DOCUMENT RESUME

ED 300 059 JC 880 473

| | |
|---|---|
| TITLE | Software Development Program. |
| INSTITUTION | Collin County Community Coll., McKinney, TX. |
| SPONS AGENCY | Texas Higher Education Coordinating Board, Austin. |
| PUB DATE | Jun 88 |
| NOTE | 101p. |
| PUB TYPE | Guides - Non-Classroom Use (055) -- Reports - Descriptive (141) -- Tests/Evaluation Instruments (160) |
| EDRS PRICE | MF01/PC05 Plus Postage. |
| DESCRIPTORS | *Associate Degrees; Community Colleges; *Computer Science Education; *Computer Software; Course Content; *Curriculum Development; *Job Analysis; *Minimum Competencies; Occupational Surveys; Program Descriptions; Programing Languages; Technical Occupations; Two Year Colleges; Vocational Education |

ABSTRACT

The Software Development Program described in this report was developed at Collin County Community College as a pilot associate degree program for the State of Texas. Drawing from an employer needs assessment and a DACUM (Developing a Curriculum) process, the two-year associate in applied science (AAS) program was developed to train scientific programmers in the competencies required to enter the job market as entry-level software engineers. Chapter 1 presents the methodology and results of three needs assessment surveys of local scientific development companies. Graphs are included, showing employers' rankings of the value of coursework in specific programming and technical skills. Chapter 2 begins with an outline of the DACUM process, which uses industry input to identify the competencies required of a specific occupation. This chapter lists the duties and tasks of a software engineer, provides the forms administered to selected workers at various companies to validate the importance of specific tasks to their job, and presents DACUM charts illustrating job duties and associated tasks. Chapter 3 contains an outline of the course requirements for an AAS in Engineering Technology and a semester-by-semester plan for completing the degree in two years. In addition, catalog course descriptions and more detailed course outlines and learning objectives are provided for each of the degree requirements. (AAZC)

**********
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
**********

ED300059

# SOFTWARE DEVELOPMENT PROGRAM

COLLIN COUNTY COMMUNITY COLLEGE
JUNE 1988

JC 880 473

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

D. L. Ray

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."



BEST COPY AVAILABLE

# ACKNOWLEDGEMENT

The Software Development Program at Collin County Community College is a pilot program for the State of Texas and was funded by the Texas Higher Education Coordinating Board through the Carl D. Perkins Act with a Program Improvement Grant.

**Project Administration:**

Bob Day, Grants Manager, Texas Higher Education Coordinating Board.

Elizabeth H. Henry, Project Administrator, Associate Vice President for Institutional Advancement, Collin County Community College.

Curtis Ward, Project Director, Program Coordinator for Software Development, Collin County Community College.

**Administrative Assistance:**

Steve Ellis, Vice President of Instruction, Collin County Community College.

Lillie Palmer, Dean of Business and Industry, Collin County Community College.

Tony McRae, Associate Dean of Business and Industry, Collin County Community College.

Gerry Perkus, Director of Institutional Research, Collin County Community College.

Sue Furnas, Division Secretary, Business and Industry, Collin County Community College.

Ken Foley, Grant Research Assistant, Collin County Community College.

Diana Weddle, Grant Secretary, Software Development Program, Collin County Community College.

**Special Thanks:**

Ron Kern, DACUM Facilitator, Business and Industry, Collin County Community College.

Diane Spears, Computer Lab Coordinator, Business and Industry, Collin County Community College.

**Equipment:**

PC's Limited 386 computer, HP LaserJet Series II printer, Ventura Desktop Publishing, WordPerfect v4.2, and Harvard Graphics v4.0. The master copy for reproduction of this manual was produced in-house by Curtis Ward and Diane Spears.

**Author:**

Curtis Ward, BS, MS, Computer Science, Southwest Texas State University, PhD student, Computer Science, University of North Texas. Taught Computer Science for two and a half years at Southwest Texas State University before joining the faculty at Collin County Community College in August, 1987. Worked in the oil industry in Texas, Alaska, and South America prior to receiving Masters degree.

# PREFACE

The Software Development Program is an Associate of Applied Science 2 year degree program designed to produce a trained scientific programmer capable of all the competencies required to enter the job market as an entry level software engineer. The program outlined in this manual meets or exceeds the validated competencies required as obtained through the DACUM procedure. Competencies are linked to particular courses but in many cases are taught in several courses at various times.

Taught correctly, this program is a challenge for the student and retention will perhaps be lower than for most other programs. However, the chances for employment at a relatively high pay scale is very good for the graduate of this program who is willing to move to a metropolitan high tech area.

Starting the program requires support from both the college and industry to make the program succeed. Small classes must be allowed to make so the program can grow. Industry can funnel students to the program and in some cases must be willing to guarantee students in order for the classes to make. Goals of the students must be observed carefully as a number of students will enroll in a few classes to learn a desired skill and then not return to school after obtaining that skill. In such cases, students are not really non-completers, but rather are students who have met their goal.

If the college and industry work together, the program can succeed and prosper. The college gains students and industry gains trained workers in a critical area. Both benefit from cooperation.

# Table of Contents


**CHAPTER 1: NEEDS ANALYSIS FOR SOFTWARE DEVELOPMENT PROGRAM . . . 1-1**

ADA PROGRAMMING SURVEY STATISTICS ..... 1-15

ASSEMBLY LANGUAGE PROGRAMMING SURVEY STATISTICS ..... 1-16

COLLEGE ALGEBRA SURVEY STATISTICS ..... 1-17

C & UNIX PROGRAMMING SURVEY STATISTICS ..... 1-18

DATA STRUCTURES SURVEY STATISTICS ..... 1-19

FORTRAN-77 PROGRAMMING SURVEY STATISTICS ..... 1-20

GROUP DYNAMICS/PROBLEM SOLVING SURVEY STATISTICS ..... 1-21

PASCAL PROGRAMMING SURVEY STATISTICS ..... 1-22

PHYSICS I SURVEY STATISTICS ..... 1-23

REAL TIME PROGRAMMING SURVEY STATISTICS ..... 1-24

SOFTWARE ENGINEERING SURVEY STATISTICS ..... 1-25

SOFTWARE TEST TECHNIQUES SURVEY STATISTICS ..... 1-26

SPEECH COMMUNICATIONS SURVEY STATISTICS ..... 1-27

TECHNICAL WRITING SURVEY STATISTICS ..... 1-28

TRIGONOMETRY SURVEY STATISTICS ..... 1-29

**CHAPTER 2: DESIGN COMPETANCY BASED CURRICULUM . . . . . 2-31**

DACUM ..... 2-35

SOFTWARE DEVELOPMENT TASK INVENTORY ..... 2-41

DACUM CHARTS ..... 2-49

**CHAPTER 3: DEGREE PLANS, COURSE DESCRIPTIONS, AND COURSE COMPETENCIES . . . . . 3-53**

ASSOCIATE OF APPLIED SCIENCE IN ENGINEERING TECHNOLOGY ..... 3-54

ASSOCIATE OF APPLIED SCIENCE ..... 3-55

CSFT 121 COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING ..... 3-60

CPSC 190 PROGRAMMING CONCEPTS I ..... 3-62

CPSC 191 PROGRAMMING CONCEPTS II ..... 3-63

CSFT 221 SOFTWARE ENGINEERING ..... 3-65

CSFT 236 INTRO TO ARTIFICIAL INTELLIGENCE ..... 3-67

CPSC 292 SCIENTIFIC PROGRAMMING ..... 3-69

CPSC 294 C PROGRAMMING LANGUAGE ..... 3-71

CSFT 223 REAL TIME PROGRAMMING ..... 3-73
CSFT 224 SOFTWARE TEST TECHNIQUES ..... 3-74
CIS 262 CUMPUTER GRAPHICS SYSTEMS ..... 3-75
CSFT 232 ADVANCED SOFTWARE ENGINEERING ..... 3-76
CSFT 233 ADVANCED ASSEMBLY LANGUAGE PROGRAMMING .... 3-78
CIS 270 NETWORKING & TELECOMMUNICATIONS ..... 3-79
CSFT 235 LISP PROGRAMMING ..... 3-80
ELEC 131 BASICS OF ELECTRONICS ..... 3-81
ENGL 151 COMPOSITION/RHETORIC I ..... 3-82
MATH 181 COLLEGE ALGEBRA ..... 3-83
PSYC 121 APPLIED PSYCHOLOGY ..... 3-84
SPCM 151 FUNDAMENTALS OF SPEECH CUMMUNICATIONS ..... 3-85
ECON 121 INTRODUCTION TO ECONOMICS ..... 3-86
HUM 151 INTRODUCTION TO HUMANITIES ..... 3-87
MATH 182 TRIGONOMETRY ..... 3-88
ENGL 291 TECHNICAL WRITING ..... 3-89

# CHAPTER ONE

# CHAPTER 1: NEEDS ANALYSIS FOR SOFTWARE DEVELOPMENT PROGRAM

1. General Survey.
    1.1. Build data base of local scientific software development companies.
        1.1.1. Buy mailing from mailing list reseller company.
        1.1.2. Obtain mailing list from local chamber of commerce.
        1.1.3. Search newspaper want-ads for companies hiring software developers.
        1.1.4. Use local contacts with high technology industry.
        1.1.5. Use in-house mailing lists.
        1.1.6. Use common knowledge.
    1.2. Design survey instrument.
        1.2.1. Modify Page 1-8 to suit local needs.
            1.2.1.1. Add line for company name and address.
            1.2.1.2. In question 1, change "unlikely" to "will not."
        1.2.2. Write cover letter similar to Page 1-7.
            1.2.2.1. Should be on letterhead.
            1.2.2.2. Program coordinator phone number should be included.
    1.3. Print mailing labels from data base.
    1.4. Mail survey instrument, cover letter, and prepaid return envelope.
    1.5. Collect returned instruments in safe place.
        1.5.1. Normally less than 20% are returned.
        1.5.2. Keep envelope with instrument.
        1.5.3. Make copies of instrument and envelope.
        1.5.4. Use copies as working set.
    1.6. Cull the instruments for unusable forms.
    1.7. Using good statistical style, perform statistical analysis on the obtained data.

1.7.1. Enlist the help of local statistics teacher or research office personnel.

1.7.1.1. If such assistance is unavailable, obtain text on educational statistics.

1.7.1.2. Use common sense.

1.7.2. Use local research company personnel if necessary.

1.8. Analyze the statistics.

1.8.1. Make careful estimates based on results.

1.8.1.1. A number of important companies possibly did not return instrument.

1.8.1.1.1. Include in secondary survey.

1.8.1.1.2. Try to make contact with software managers of important companies.

1.8.1.2. Results should not be expanded in direct proportion to percentage of returned instruments.

1.8.1.2.1. Too many variables to allow direct proportion expansion.

1.8.1.2.2. Be conservative in estimates of need.

1.8.2. Estimate number of jobs available to AAS degree graduates.

1.9. Sort returned instruments into two groups.

1.9.1. Group one.

1.9.1.1. Companies answering "might' and "likely" to question 1.

1.9.1.2. Companies showing scientific languages in question 6.

1.9.1.3. Companies writing favorable comments.

1.9.2. The remaining companies.

1.10. Build data base of companies in the first sorted group.

2. Secondary survey.

2.1. Build data base of companies not responding to first survey.

2.1.1. Add any additional companies not in first data base.

2.1.2. Make mailing labels.

2.2. Type cover letter for secondary survey with personal plea to please respond.

2.3. Adjust survey instrument if necessary.

2.3.1. Something missing.

2.3.2. Instructions unclear.

2.3.3. Focus incorrect.

2.4. Mail secondary survey

2.4.0.1. Cover letter.

2.4.0.2. Survey letter.

2.4.0.3. Prepaid return envelope.

2.5. Execute process for returned survey as above.

2.6. Add sorted first group companies to data base of sorted first group companies from general survey.

3. Follow-up interviews.

3.1. Obtain personal industry viewpoint.

3.1.1. Telephone contact person from sorted data base.

3.1.1.1. Develop relationship with contacts.

3.1.1.2. Attempt to determine real needs.

3.1.1.3. Determine feelings toward two year degree.

3.1.2. Visit "good" contacts.

3.1.2.1. Continue to develop relationship.

3.1.2.2. Determine what college can do for the company.

3.1.2.3. Determine what company can do for the college.

3.2. Through personal contacts, select members of advisory committee

3.2.1. No more than one or two persons from any one company.

3.2.2. Good mix of large, medium, and small companies.

3.2.3. Mid-level management personnel probably the best.

3.2.3.1. Too high, have lost contact with actual needs.

3.2.3.2. Too low, have no power to get things done.

3.2.4. Preferably computer science degree.

3.2.5. Preferably strong person but not domineering.

3.2.5.1. Common sense.

3.2.5.2. Willing to compromise.

3.2.5.3. Willing to listen.

3.2.6. Committee size: 10-20.

# COLLIN COUNTY COMMUNITY COLLEGE

## OFFICE of INSTITUTIONAL RESEARCH

## SURVEY of SOFTWARE DEVELOPMENT COMPANIES: EXECUTIVE SUMMARY

1. In the Winter of 1988, surveys were mailed to 575 computer software development related companies in Collin, Tarrant and Dallas counties. Seventy (70) companies returned completed surveys providing job descriptions, duties and tasks for employees that work in computer software development. The findings from the survey will help CCCC determine course content for the software development curriculum.

Companies surveyed employ software development personnel in various areas including: engineering, business data processing, consulting and scientific/technical applications. Two response modes were used in the questionnaire design: (1)forced-response questions (rank order and nominal scale) and (2) open-ended questions. Of the responding companies, 67% indicated they would be most likely to or might hire an individual with an associate degree in Computer Software Development. This perhaps could indicate a general willingness of companies to hire individuals who obtain computer competency skills through a concentrated two year degree program.

The range of responses placed computer theory and applied courses as most important, specifically: Software Engineering, Software Test Techniques, Data Structures, C + Unix programming and Real Time programming. In addition, there seemed to be significant interest that employees take courses teaching fundamental communication skills, specifically: Group Dynamics/Problem Solving, Technical Writing and Speech Communications. From this statistical analysis, it appears companies want employees to emerge from a program with a mix of fundamental, theoretical and applied skills related to day to day work.

The particular job descriptions, tasks and duties as specified by responding companies are presented in the Executive Summary. Many companies commented that new employees need to be a productive component from the start, applying basic techniques for software solutions.

While the associate degree in Software Development is aimed at preparing individuals for scientific/technical positions, an individual who emerges from the program can cross into other related fields with little trouble, for example business software development. The data obtained in the survey may be swayed by companies that have operations centered on a business format. Thus the data may indicate a stronger background in accounting and office procedures are needed.

Of the responding companies, 85% indicated they may hire 1-2 individuals per year who would graduate with an associate degree. From analysis of data it appears the demands of employers for applied programming in 'C' language on a IBM-PC compatible or mainframe, like a Digital Vax, will be satisfied by Software Development graduates.

# Collin County Community College District

January 29, 1988

Dear Business Man or Woman:

The enclosed brief questionnaire is part of a major study Collin County Community College (CCCC) is undertaking in order to obtain valid competency requirements for entry-level technical jobs in software development. CCCC is establishing an associate degree program in Engineering Technology with a major in Software Design. This program is the first in the state to be aimed specifically at educating individuals for software development in companies such as yours. Based on the information we receive from this questionnaire, CCCC can more effectively prepare potential employees for developing programs and updating software.

Your response is extremely important to our study. All responses will be treated confidentially and will be used only in group analyses. A prepaid envelope is included for your convenience. Please fill out and return this questionnaire by February 15 (or redirect to the proper staff person). Thank you for your assistance!

Sincerely,

Curtis D. Ward
Coordinator,
Software Design Program

# COLLIN COUNTY COMMUNITY COLLEGE

## BUSINESS SURVEY:
## SOFTWARE TECHNOLOGY PERSONNEL NEEDS

(1) In general, are you or your company likely to employ an individual with a two-year associate degree in Computer Software Development for software design and/or as a programmer?

____ likely
____ might
____ unlikely

(2) If you would employ such graduates, how many individuals might you hire per year?

____ 1-2
____ 3-10
____ 11 +

(3) What are the title(s) of such position(s) to be filled?

1) _____
2) _____
3) _____

(4) The typical starting annual salary for such a position is in the range of ...

____ $15,000 - $18,000 ____ $18,000 - $21,000
____ $21,000 - $24,000 ____ $24,000 +

(5) Please give a brief job description for such position(s).

_____
_____
_____
_____

(6) Below are courses which CCCC is considering offering for its Associate of Applied Science Degree Program in Engineering Technology with a major in Software Design. Please rate each course from 1 to 5 according to its relative importance for training an entry-level employee in your company (5 = most important, 1 = least important).

____ Assembly Language Programming
____ Data Structures
____ C + Unix Programming
____ Real Time Programming
____ Group Dynamics/Problem Solving
____ Technical Writing
____ Physics I
____ Trigonometry
____ other _____

____ Pascal Programming
____ Ada Programming
____ Fortran 77 Programming
____ Software Test Techniques
____ Software Engineering
____ Speech Communications
____ College Algebra
____ other _____
____ other _____

For the items in each of the following questions, give the percentage (out of 100%) that your company would allot to each item as it is used for development purposes.

(7) Type of computer(s).

____% IBM-PC compatible ____% Digital VAX
____% IBM 360-370 ____% _____

(8) Operating system(s).

____% MS-DOS, PC-DOS ____% VMS
____% Unix (Ultrix, etc.) ____% _____

(9) Operative language(s).

____% Pascal ____% C
____% Assembly ____% Fortran
____% Basic ____% _____

(10) If your company employs CAD, Electronics or Software Development Technicians please list a contact person and telephone number for each of these areas.

CAD_____ ph#_____
Electronics_____ ph#_____
Software Development_____ ph#_____

Please give any additional comments or suggestions:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

**An Equal Opportunity Institution**
**Office of Institutional Research**
**Jan. 1988**

# COLLIN COUNTY COMMUNITY COLLEGE
# OFFICE OF INSTITUTIONAL RESEARCH
# SOFTWARE TECHNOLOGY SURVEY
# SPRING 1988: SUMMARY OF RESULTS

In the Spring of 1988, companies in Collin, Tarrant, and Dallas counties were surveyed to obtain competencies required for entry and mid-level jobs in software development. Responding companies returned completed surveys providing job descriptions, duties and tasks for employees that work in computer software development. Information obtained from respondents will help CCCC to determine course content for the software development curriculum. Of the 575 companies surveyed, 70 or 13% of the total population returned a completed survey. The following is a summary of survey results:

*In questions 1 and 2, the demand is established for CCCC students who would graduate with an associate degree in Software Development.*

(1) In general, are you or your company likely to employ an individual with a two-year associate degree in Computer Software Development for software design and/or as a programmer?

22% likely 45% might 33% unlikely

(2) If you would employ such graduates, how many individuals might you hire per year?

85% 1-2 15% 3-10

*Questions 3 through 5 pinpoint the title(s), salary and job description for such position(s). Forty responding companies provided information about these items. For purposes of analysis, positions in software development are classified into two categories, Business and Scientific. The number of companies expressing a demand for specialists at a particular salary range is indicated in the chart below.*

**Scientific**

Programmer/Analyst-two levels of expertise

Software Technician

Analyzes problems, plans and develops software programs. Enters coded commands into computer. Corrects errors by altering commands. Instructs computer operators on data input to obtain required results.

| Salary Range | Demand |
|---|---|
| $18k-21k | 7 companies |

Software Engineer

Designs, modifies and installs system software programs and hardware for problem solution. Helps guide the activities of the software technicians and computer operators in developing software programs. Submits data specifications on chip requirements to the engineering department.

| Salary Range | Demand |
|---|---|
| $18k-21k | 2 companies |
| $21k-24k | 13 companies |
| $24k | 3 companies |

**Business**

Programmer/Trainee

Applies knowledge of computer capabilities to develop sequence of program steps for solution of business problems. Works with departmental representatives to determine program intent and out- put requirements. Works as part of a team to design and write programs; debugs programs. Writes documentation for programs. Devises and maintains data processing systems. May coordinate and direct the activities of other workers.

| Salary Range | Demand |
|---|---|
| $15k-18k | 11 companies |
| $18k-21k * | 1 companies |
| $21k-24k ** | 3 companies |

* Program design in RPG III
** Program design in Cobol language

*In question 6, courses are listed which CCCC is considering offering for its Associate of Applied Science Degree Program in Engineering Technology with a major in Software Design. Respondents were requested to rate each course from 1 to 5 according to its relative importance for training an entry-level employee for their company (5 = most important, 1 = least important). The results of these ratings are reported below. In addition to the mean, the mode and median for each data set are presented so as to provide a comprehensive measure of central tendency.*

| TENTATIVE COURSES | *MEAN | **MODE | ***MEDIAN |
|---|---|---|---|
| Software Test Techniques | 3.73 | 4,5 | 4 |
| Software Engineering | 3.71 | 5 | 4 |
| Data Structures | 3.62 | 5 | 3 |
| C + Unix Programming | 3.23 | 5 | 3 |
| Group Dynamics/Problem Solving | 3.20 | 4 | 4 |
| Technical Writing | 3.17 | 3 | 3 |
| Assembly Language Programming | 3.14 | 5 | 3 |
| Real Time Programming | 3.00 | 3 | 3 |
| College Algebra | 2.88 | 1 | 3 |
| Pascal Programming | 2.12 | 1 | 2 |
| Trigonometry | 1.92 | 1 | 1 |
| Fortran Programming | 1.76 | 1 | 1 |
| Physics I | 1.64 | 1 | 1 |
| Ada Programming | 1.54 | 1 | 1 |

* Mean = the average of all measurements in a set of data
** Mode = the most frequently occurring measurement in a set of data
*** Median = the middle measurement in a set of data

The ten courses most frequently named by companies in the "other" category are: Cobol and RPG III language programming, Accounting, Basic and Mumps language programming, System Design, Personal Computers, Calculus, Database Theory, Data Communications.

*In questions 7-9, each company indicated the percentage (out of 100%) that it allocates to particular types of computers, operating systems and operative languages that are used for development purposes. The number of companies citing a particular development tool is indicated in the chart below. Also indicated is the average percentage of use allotted to that particular development tool by users.*

Computers

| Type of computer | Number of users | Average percentage of use |
|---|---|---|
| IBM-PC compatible | 38 | 53.3% |
| Digital VAX | 8 | 65.0% |
| IBM 360-370 | 12 | 49.6% |

Given additional space for comment, companies indicated other computer(s) which they use. The most frequent of these in order are: IBM 5/38 and 5/36, MacIntosh and Apple II family, DEC, PDP, Alpha Micro, Unix, Honeywell, Prime.

Operating Systems

| Type of operating system | Number of users | Average percentage of use |
|---|---|---|
| MS-DOS, PC-DOS | 33 | 57.5% |
| VMS | 10 | 71.5% |
| Unix (Ultrix, etc.) | 12 | 53.8% |

In additional space for comment, companies indicated other operating system(s) which they use. The most frequent of these in order are: SSP/CPF, MVS, IBM OS & DOS/VSE, BOSS.

Operative Languages

| Type of operative language | Numbers of users | Average percentage of use |
|---|---|---|
| Pascal | 7 | 43.6% |
| 'C' | 25 | 49.6% |
| Assembly | 26 | 27.3% |
| Basic | 13 | 67.2% |

Given additional space for comment, companies indicated other operative language(s) which they use. The most frequent of these in order are: Cobol, RGP II & III, Mumps, dBaseIII, Ada.

*Additional comments made by respondents covered a range of abilities that they desire to see in prospective employees:*

"We work with robotics and are currently developing cell software products focused toward the semiconductor industry. As we grow from our current head count of twenty, we may be interested in seeing CCCC graduates. Up to now, all of our people are senior level with a minimum of five years experience."

"Prospective employees must be fluent in C language and have a good working knowledge of Unix."

"The soul of a good programmer is based on his/her ability to see the complete shape of the project, and then direct steps to build the shape. To do this, a person needs a good background in humanities, mathematics and programming in a variety of situations."

"Touch typists make better programmers."

"Very hard to find a programmer that can handle all aspects of developing an application program. Some know theory, others know languages, some know 'tricks and games' - few know it all. Most of all, few can be trusted to produce an absolutely foolproof product that doesn't cause major support problems."

"Please teach them basic reading and writing skills! I cannot describe how many candidates come to me with excellent technical credentials, but completely lack basic communication skills. If an applicant cannot write effectively, I simply will not hire them!"

"We prefer four-year degrees, but would look at a talented two-year degreed person."

"We recruit primarily from four-year universities. A sound program with a good mix of theoretical and applied is mandatory (main-frame and business applications)."

"Our products are heavily graphic oriented. Having background in graphic applications is important to us when evaluating employment candidates."

"Our firm develops and maintains software exclusively for IBM products. While we almost always have positions open, as a consulting firm, we cannot afford to hire programmers with no experience."

## GRAPH EXPLANATION

On the following pages are graphical representations of the results of three surveys.

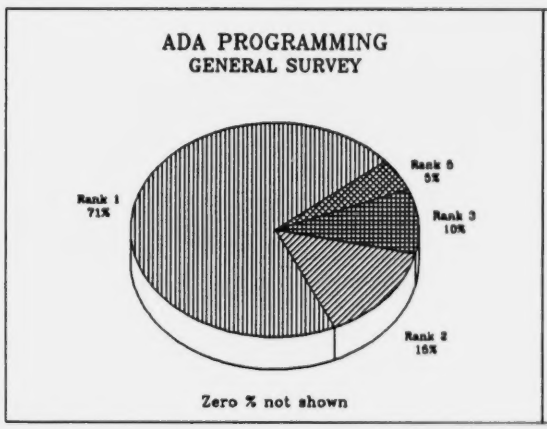
1. General Survey. This survey was conducted in the Greater Dallas area with purchased labels. Many companies surveyed had little in common with the Software Development Program.

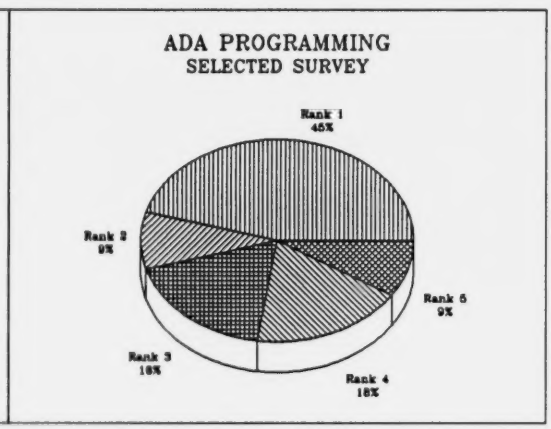
2. Selected Survey. This is a set of returned survey instruments from the General Survey, selected by observing from the answers if the company returning the survey was a company that might hire a Software Development graduate. In other words, this subset represents companies which employ scientific programmers.

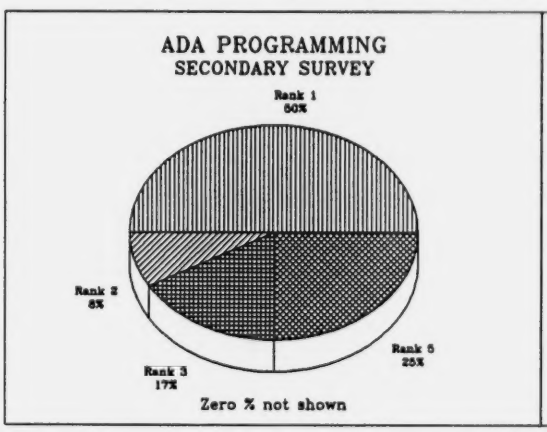
3. Secondary Survey. Since less than 20% of the companies returned instruments, there were important companies not included in the results. A search of the newspaper for ads being run by companies looking for scientific programming workers in addition to several large companies known to hire scientific programmers produced a list of companies for a secondary survey.

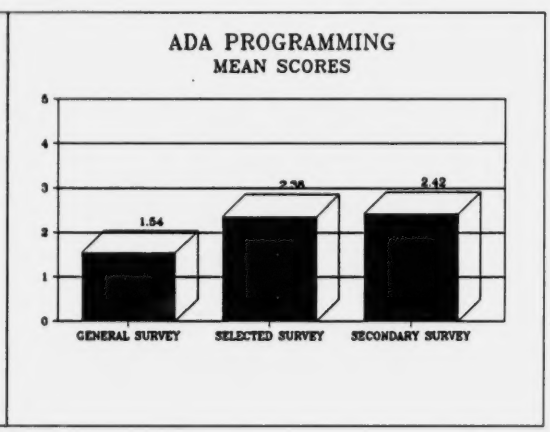
The three pie charts on each page show a ranking out of 100% where 5 is the most important and 1 is the least important. The bar chart is the average rank for that particular course for all three surveys.

# ADA PROGRAMMING
# SURVEY STATISTICS



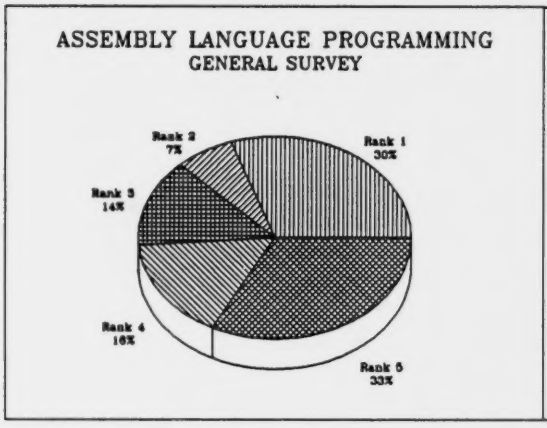



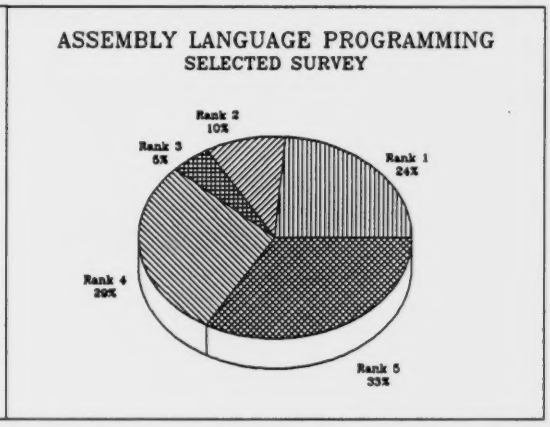



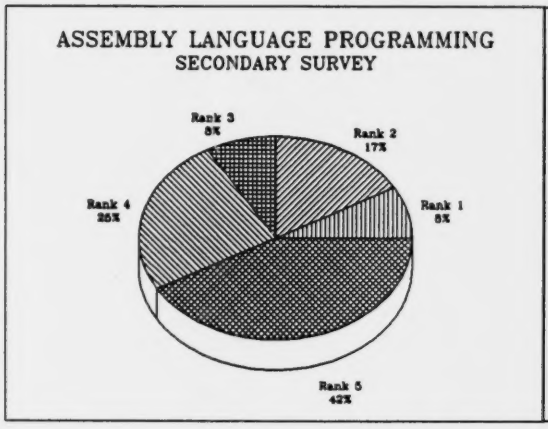



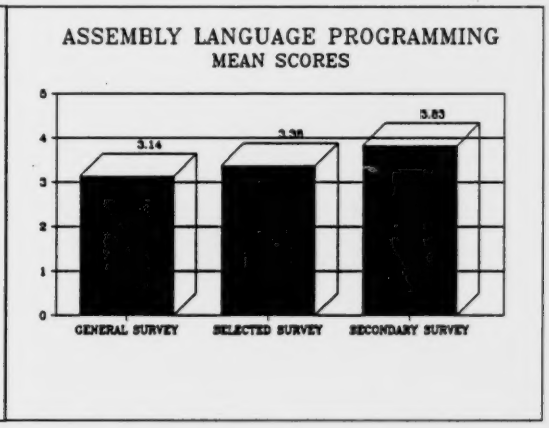

**5 = most important, 1 = least important**

# ASSEMBLY LANGUAGE PROGRAMMING
# SURVEY STATISTICS



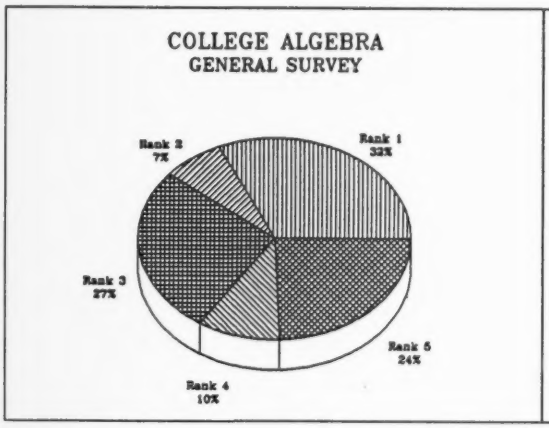



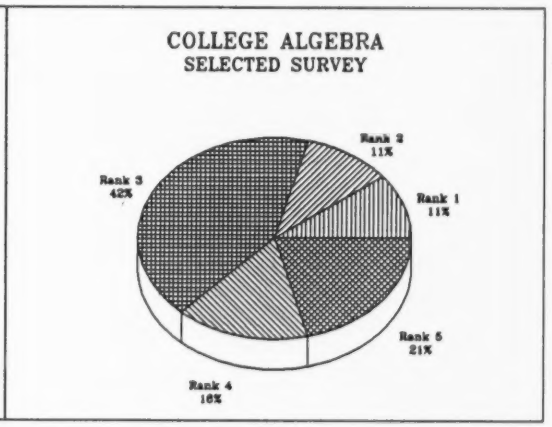



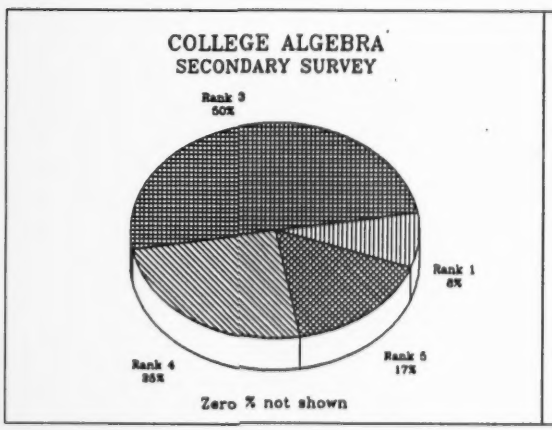



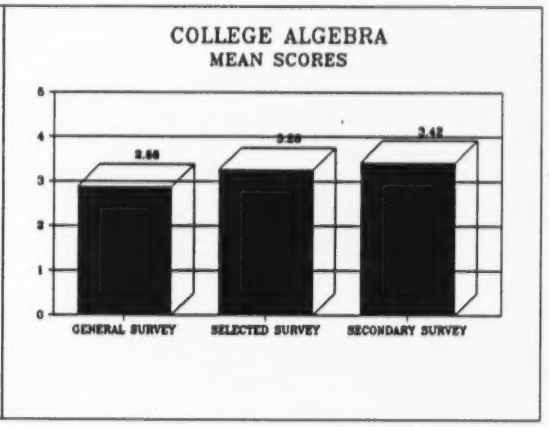

**5 = most important, 1 = least important**

# COLLEGE ALGEBRA
# SURVEY STATISTICS



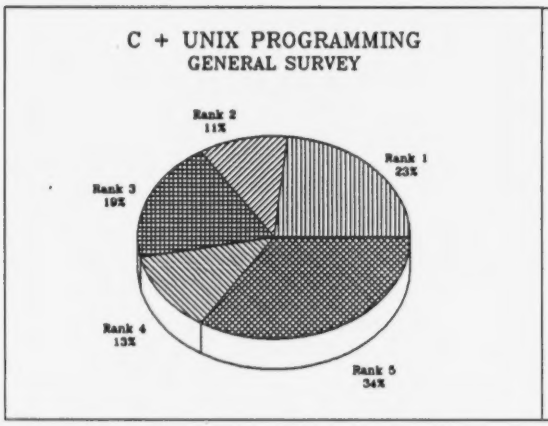



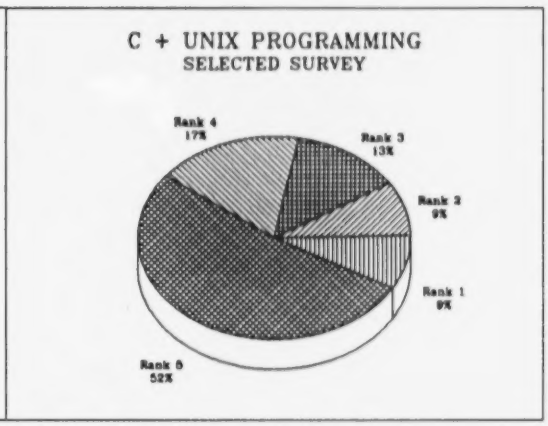



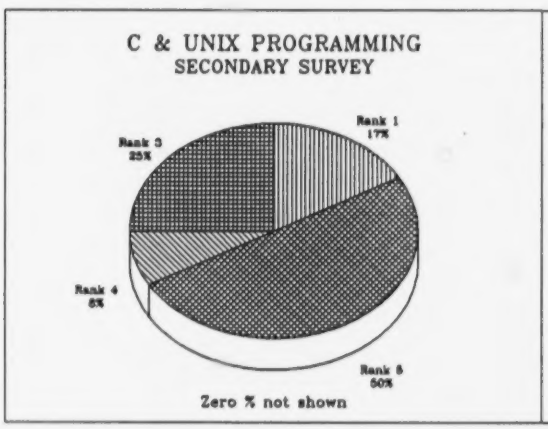



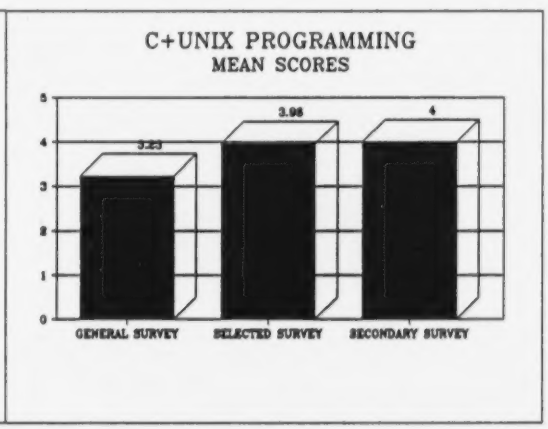

**5 = most important, 1 = least important**

# C & UNIX PROGRAMMING
# SURVEY STATISTICS



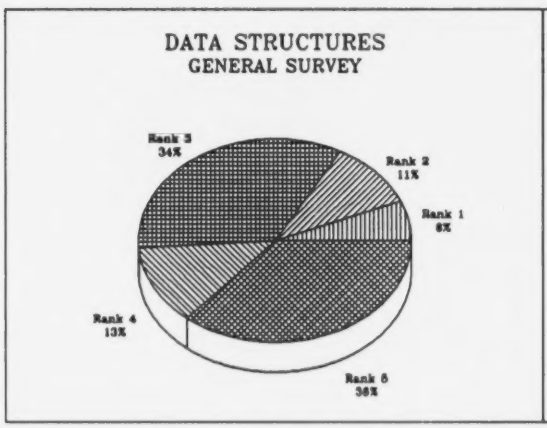



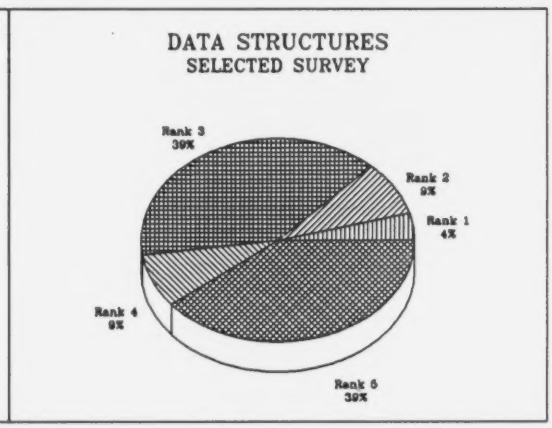



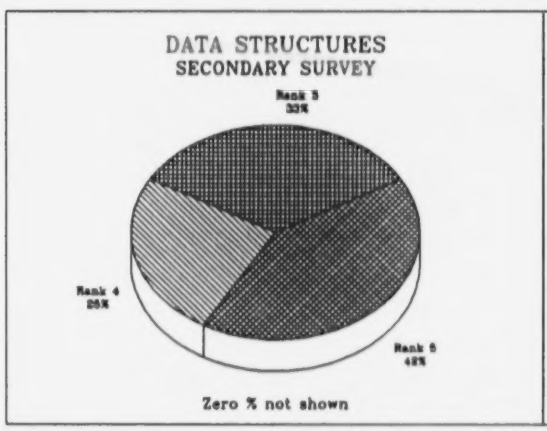



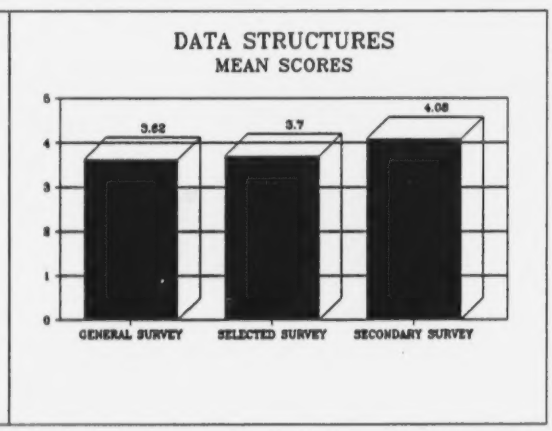

**5 = most important, 1 = least important**

# DATA STRUCTURES
# SURVEY STATISTICS



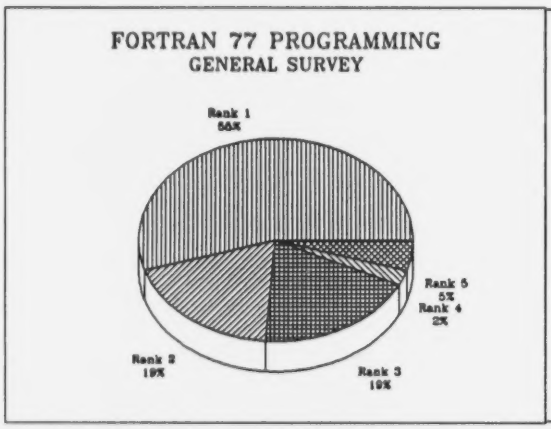



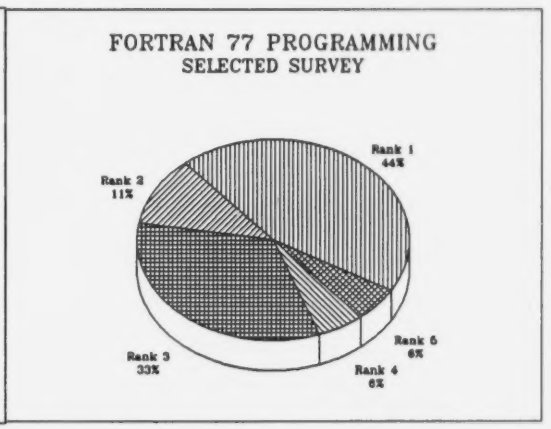



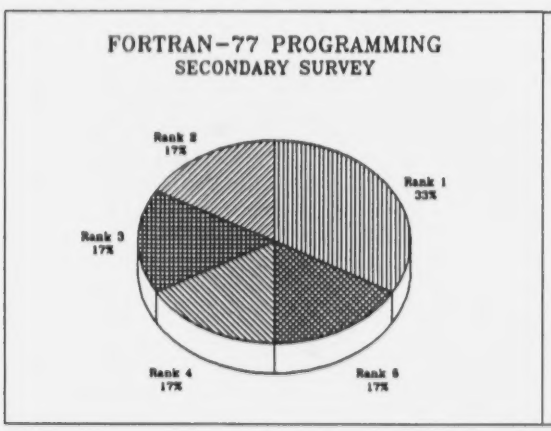



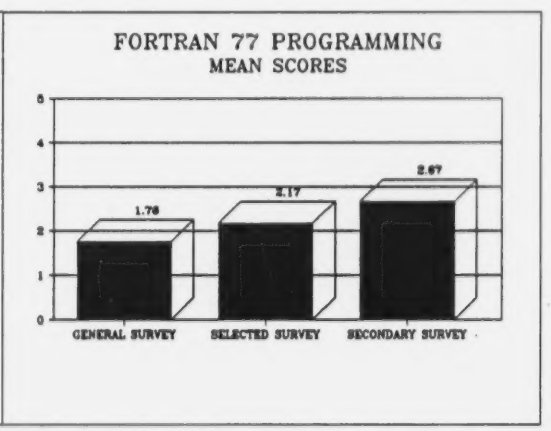

**5 = most important, 1 = least important**

# FORTRAN-77 PROGRAMMING
# SURVEY STATISTICS



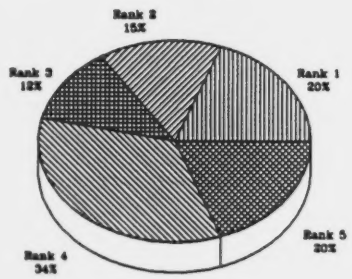



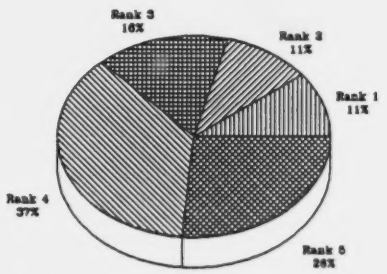

**5 = most important, 1 = least important**

# GROUP DYNAMICS/PROBLEM SOLVING SURVEY STATISTICS

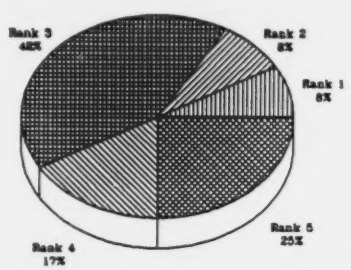
GROUP DYNAMICS/PROBLEM SOLVING
GENERAL SURVEY



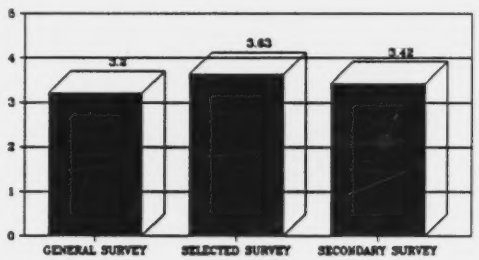

GROUP DYNAMICS/PROBLEM SOLVING
SELECTED SURVEY




GROUP DYNAMICS/PROBLEM SOLVING
SECONDARY SURVEY

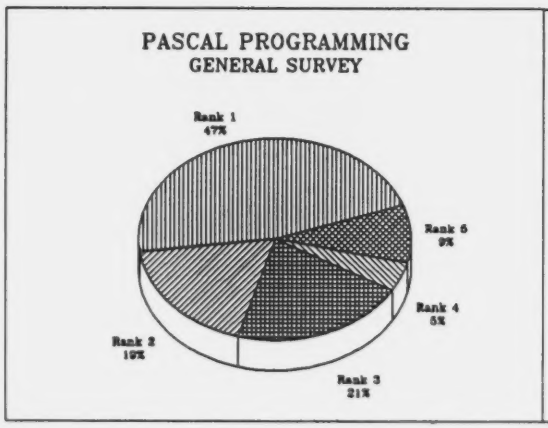


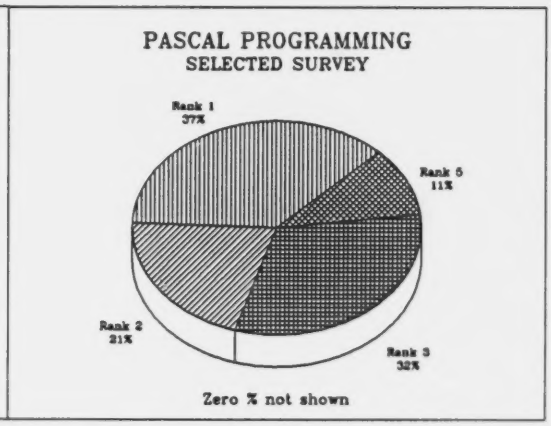

GROUP DYNAMICS/PROBLEM SOLVING
MEAN SCORES

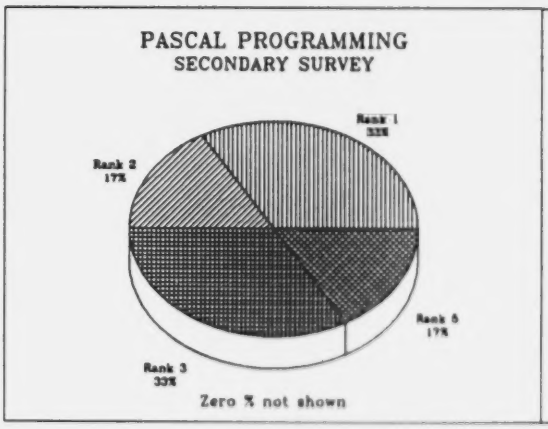


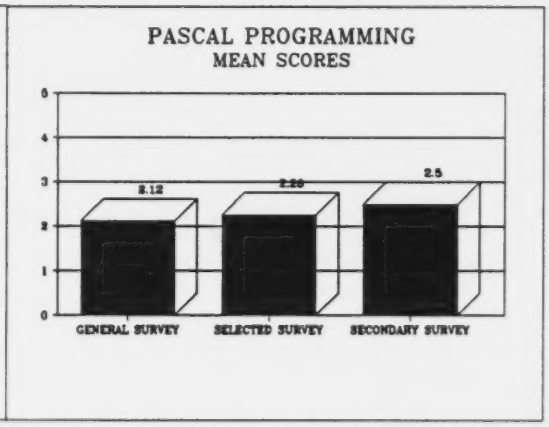

**5 = most important, 1 = least important**

# PASCAL PROGRAMMING
# SURVEY STATISTICS



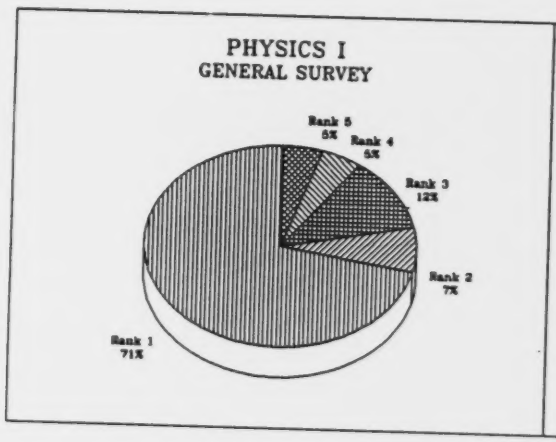



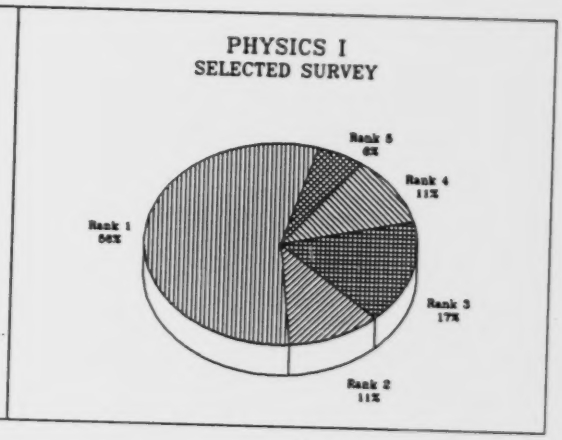



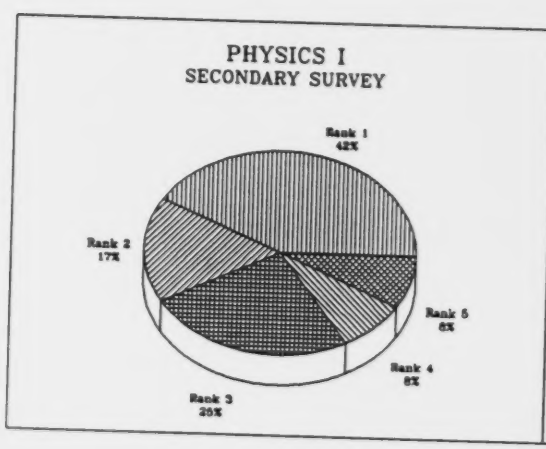



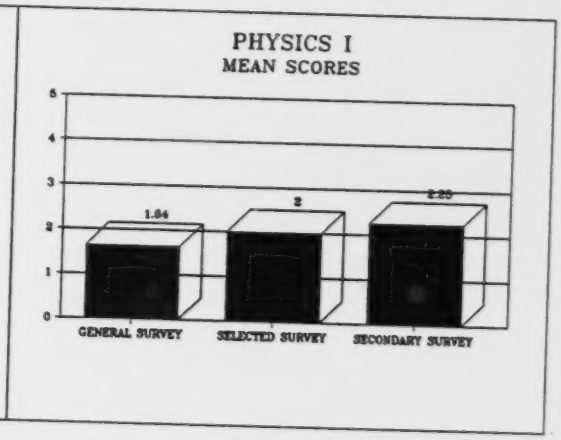

**5 = most important, 1 = least important**

# PHYSICS I
# SURVEY STATISTICS



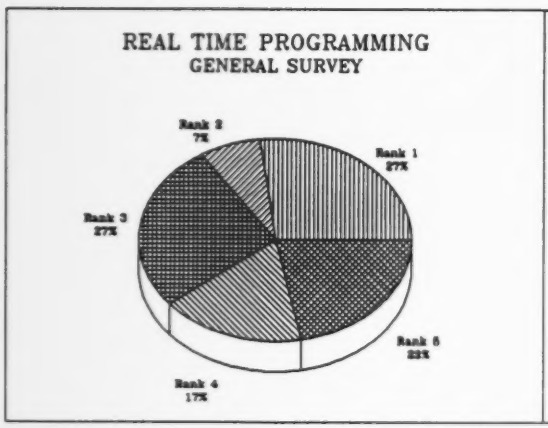



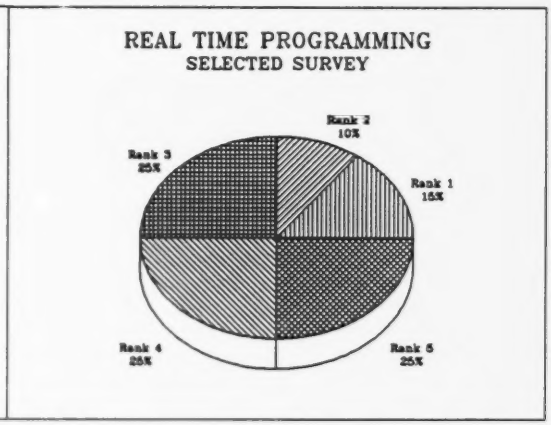



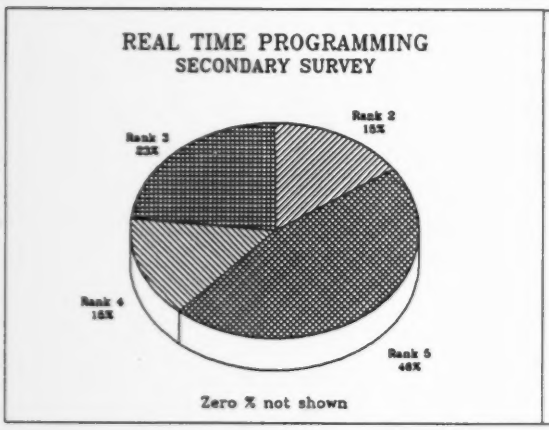



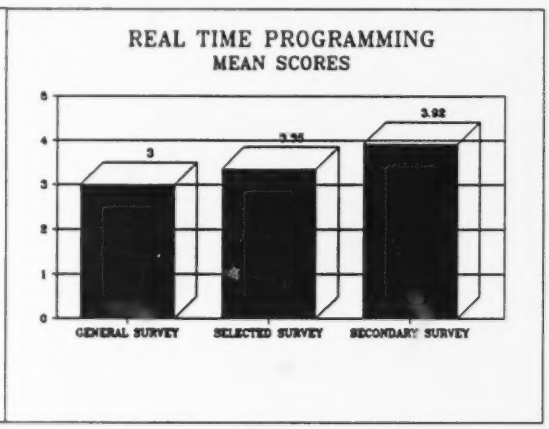

**5 = most important, 1 = least important**

# REAL TIME PROGRAMMING
# SURVEY STATISTICS



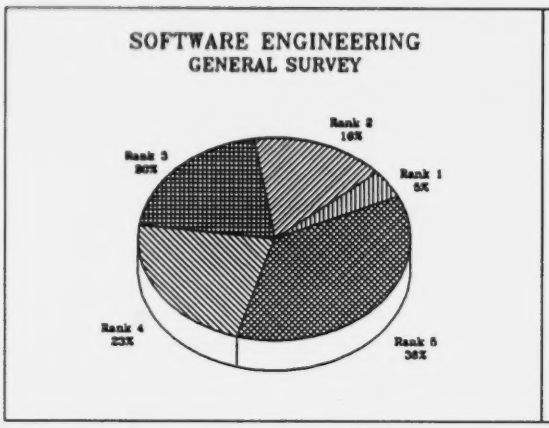



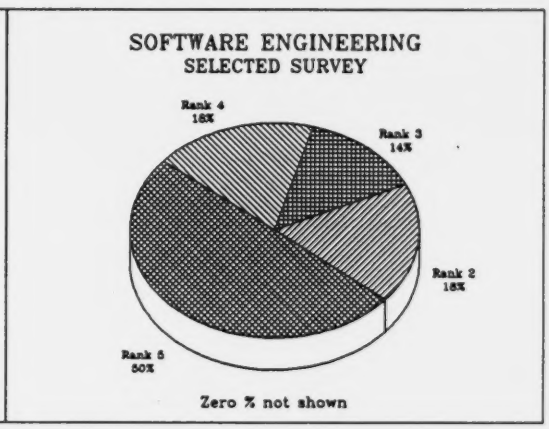



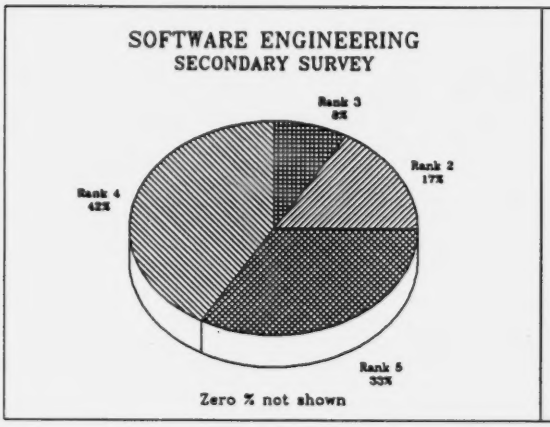



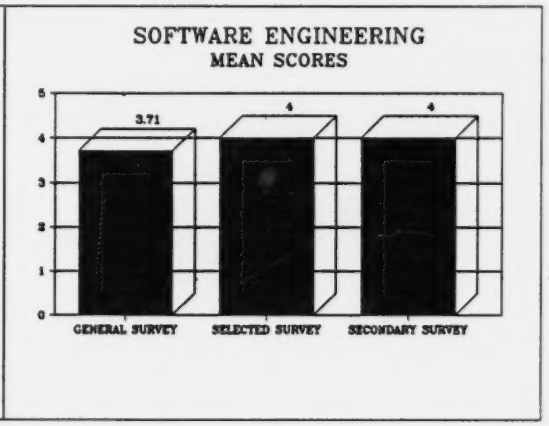

**5 = most important, 1 = least important**

# SOFTWARE ENGINEERING
# SURVEY STATISTICS



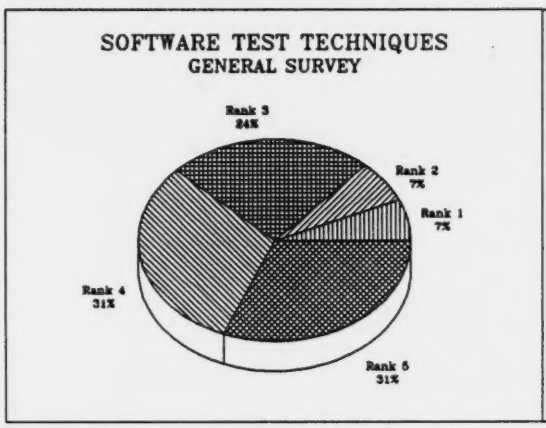



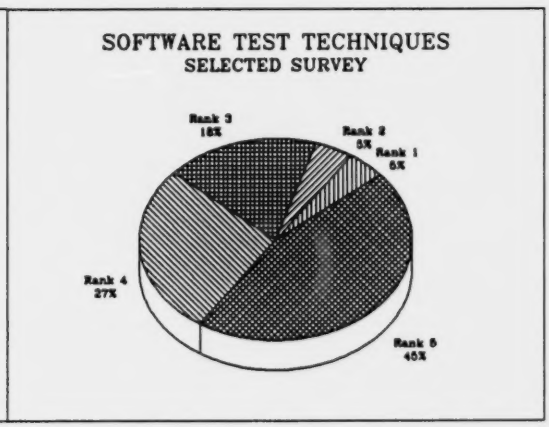



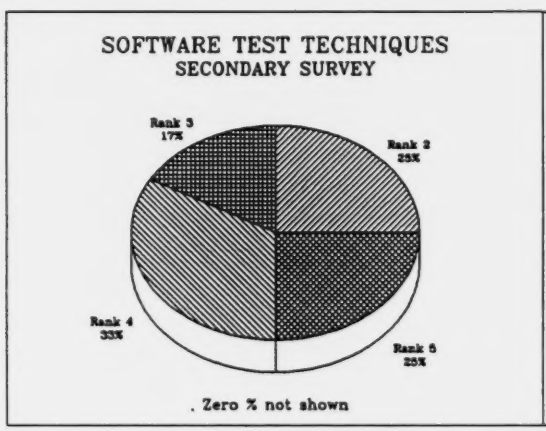



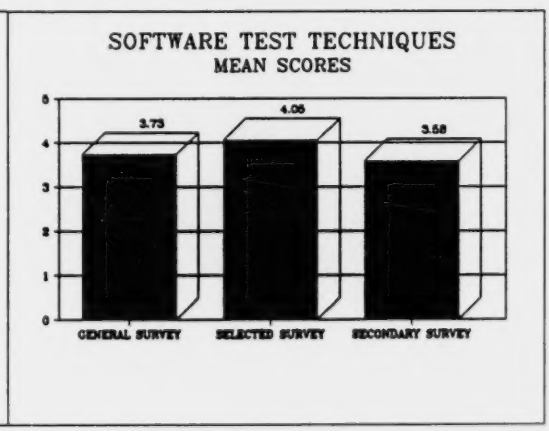

**5 = most important, 1 = least important**

# SOFTWARE TEST TECHNIQUES
# SURVEY STATISTICS



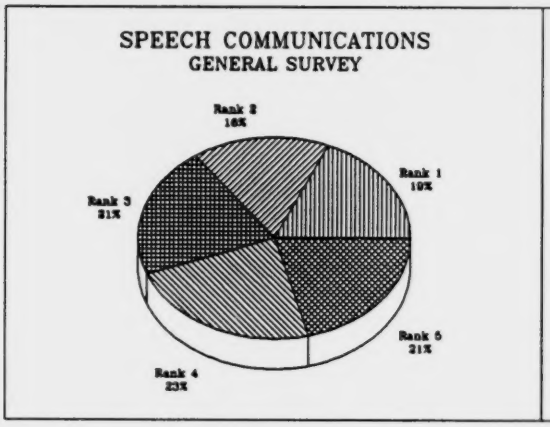



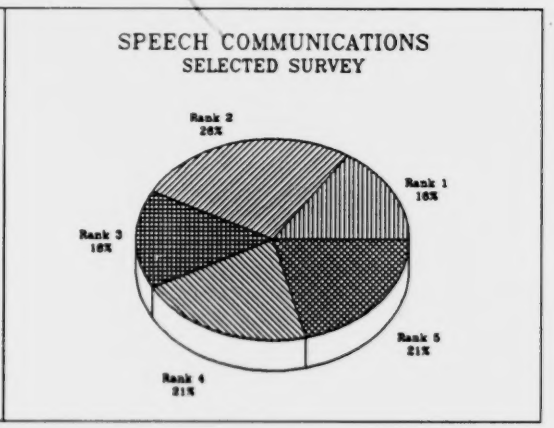



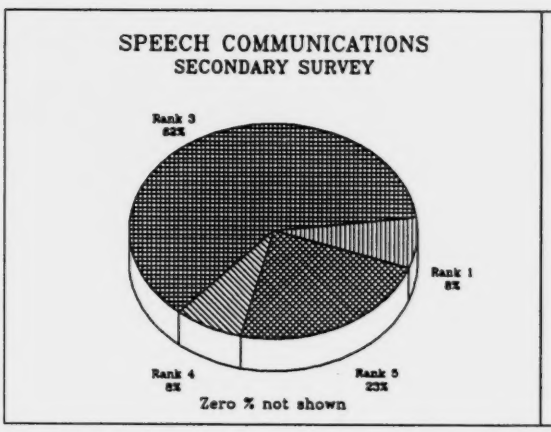



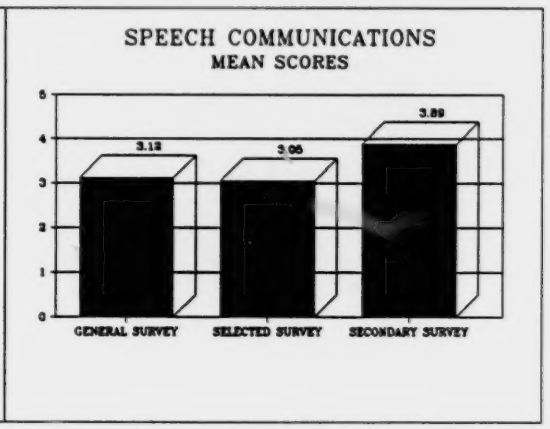

**5 = most important, 1 = least important**

# SPEECH COMMUNICATIONS
# SURVEY STATISTICS



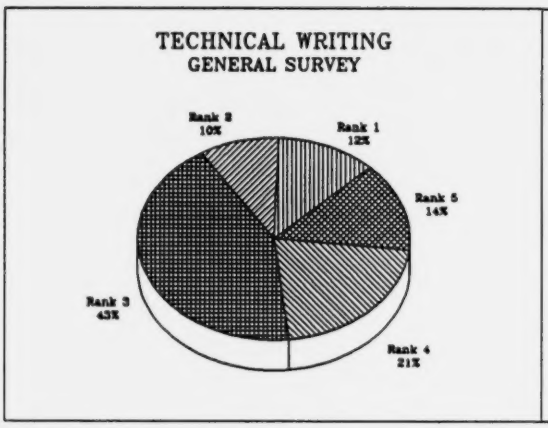



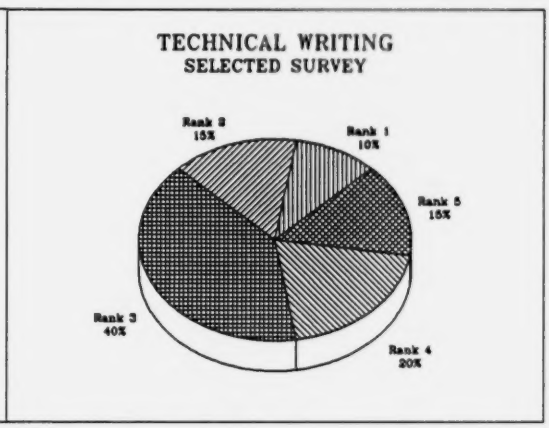



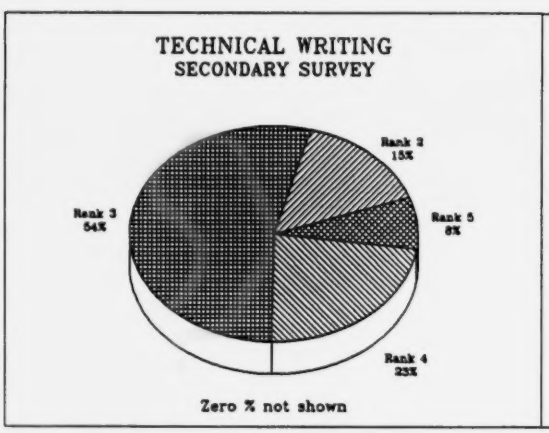



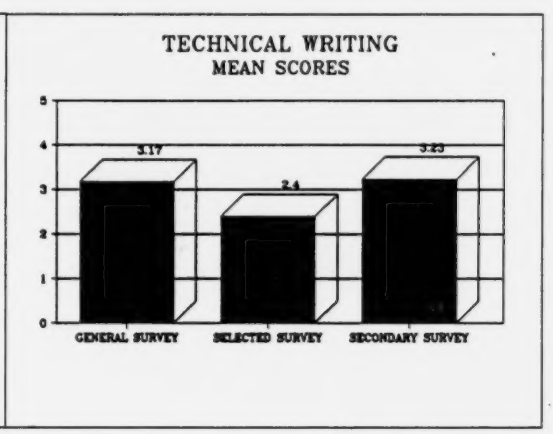

**5 = most important, 1 = least important**

# TECHNICAL WRITING
# SURVEY STATISTICS



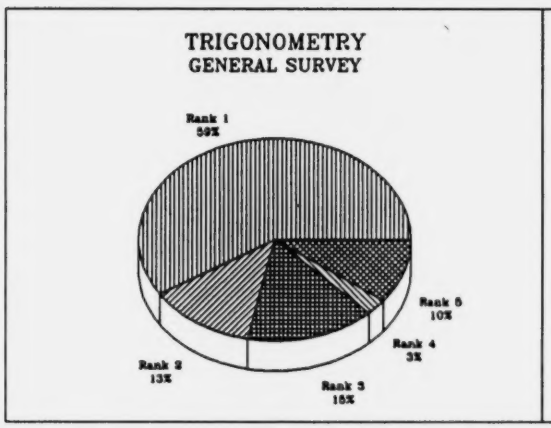



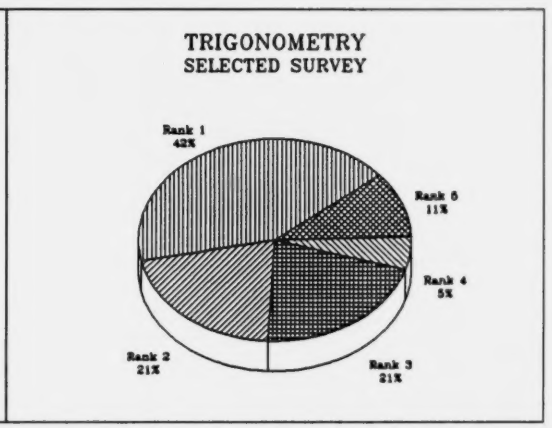



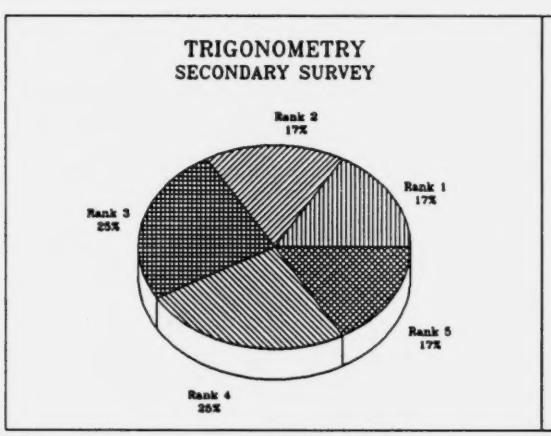



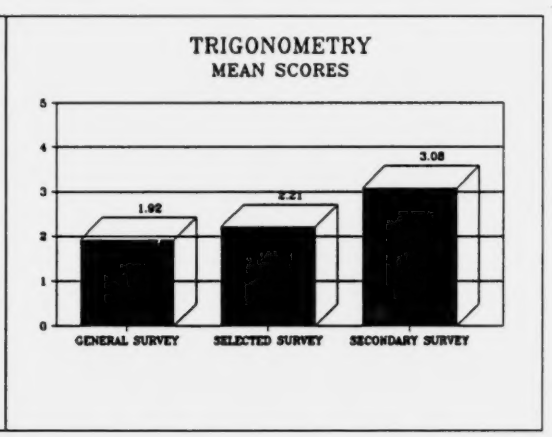

**5 = most important, 1 = most important**

# TRIGONOMETRY
# SURVEY STATISTICS




**5 = most important, 1 = least important**

# CHAPTER TWO

# CHAPTER 2: DESIGN COMPETENCY BASED CURRICULUM

1. DACUM

   1.1. Organize DACUM panel.

      1.1.1. Ask experienced fellow teacher to be DACUM facilitator.

         1.1.1.1. Not necessary to have teacher knowledgeable in computer science.

         1.1.1.2. Assist wherever posible.

         1.1.1.3. DACUM panel should not last over 2 days.

            1.1.1.3.1. Donuts and coffee.

            1.1.1.3.2. Noon meal for 2 days.

      1.1.2. Need a recorder to assist DACUM facilitator.

      1.1.3. Use DACUM Handbook from the National Center for Research in Vocational Education, Ohio State Universiy.

   1.2. Ask industry contacts to supply DACUM panel members.

      1.2.1. Need 3-4 members from supervisory positions.

      1.2.2. Need 8-12 members from entry-level positions.

         1.2.2.1. Best with at least one year in job.

         1.2.2.2. Different job titles.

      1.2.3. Get firm commitments.

         1.2.3.1. More is better up to about a total of 15.

         1.2.3.2. Getting this committment is difficult.

         1.2.3.3. Start early getting committments.

   1.3. Hold DACUM panel.

      1.3.1. Use DACUM Handbook to guide the panel.

      1.3.2. DACUM facilitator produces Final Report (Page 2-35).

   1.4. Validate DACUM competencies.

      1.4.1. Produce task inventory (Page 2-41).

1.4.2. Produce instruction sheet (Page 2-40).

1.4.3. Send to companies on sorted data base.

1.4.3.1. Produce personalized cover letter to industry contact.

1.4.3.2. Telephone contact.

1.4.3.2.1. Inform contact about validation.

1.4.3.2.2. Ask number of validation forms needed.

1.4.3.3. Enclose task inventory, instruction sheet, and prepaid envelope.

1.4.4. Analyze results.

2. Advisory committee meeting.

2.1. Agenda.

2.1.1. Address goals of the college.

2.1.2. Address goals of the software development program.

2.1.2.1. For the college.

2.1.2.2. For industry.

2.1.3. Discuss a proposed curriculum.

2.1.3.1. Use material from Chapter 3.

2.1.3.2. Be willing to compromise and adapt.

2.1.3.3. Listen to ideas from the committee on local needs.

2.1.4. Discuss support from industry.

2.1.4.1. Blocked classes.

2.1.4.2. In-house training.

2.1.4.3. Flex-time for workers to attend classes.

2.2. Need 10 to 15 members in attendance.

2.3. Provide meal.

2.4. Locate good meeting place.

MEMORANDUM

TO: Curtis Ward, Software Development Program Coordinator

FROM: Ronald P. Kern, DACUM Facilitator

DATE: May 25, 1988

SUBJECT: DACUM Panel Final Report

---

The DACUM panel for Software Development Curriculum was convened on May 24, and 25, 1988. The panel consisted of representatives whose current job positions were as follows:

1. Software Design Engineer
2. Software Technician
3. Entry-Level Software Engineer
4. Assistant Programmer

The panel was conducted in adherence to the DACUM process guidelines as outlined by The National Research Center For Research in Vocational Education at The Ohio State University. The attached report is the written conclusions of the committee.

## DACUM EXPLANATION

The following pages contain the competencies acquired by the DACUM panel.

# DACUM

## REVIEW THE OCCUPATION

JOB TITLES DESCRIBED:

Software Technician
Assistant Programmer
Assistant Software Engineer
Entry Level Programmer
Entry Level Software Engineer
Programmer

JOB DESCRIPTION:

Works with Lead Engineer, Programmer/Analyst and/or Management Users to:

1. Read and modify existing code to fit new specifications,
2. Finish coding partially written programs,
3. Code a pre-designed program to required specifications and standards, and
4. Quickly learn and apply a new software tool by reading documentation and manuals.

## DUTIES AND COMPETENCY TASKS

DUTIES TASKS SUBTASKS

1.0 BASIC COMPUTER SKILLS

- 1.1.0 Identify and Use Computer Terminology
- 1.2.0 Boot and/or Logon to a Computer
- 1.3.0 Use Operating System Commands
- 1.4.0 Manage a File
  - 1.4.1 Name a File
  - 1.4.2 Create and Use Subdirectories

2.0 COMPUTER APPLICATIONS PROGRAMS

- 2.1.0 Use a Word Processing Package
- 2.2.0 Use a Database Manager
- 2.3.0 Use an Electronic Spreadsheet
- 2.4.0 Use a Terminal Emulator

3.0 SOFTWARE DEVELOPMENT TOOLS

- 3.1.0 Use a Text Editor

3.2.0 Use an Interpreter

3.3.0 Use a Compiler

3.4.0 Use an Assembler

3.5.0 Use a Linker

3.6.0 Use a Loader

3.7.0 Use a Symbolic Debugger

3.8.0 Use Assembly Language

3.9.0 Use at Least One High-Level Language

4.0 DOCUMENTATION

4.1.0 Follow Documentation Instructions

4.2.0 Use Documentation to Learn New Software

4.3.0 Apply Cross-Referencing Techniques

5.0 SYSTEM DESIGN

5.1.0 Read and Write Pseudo Code

5.2.0 Read and Write Flow Charts

5.3.0 Read and Write HIPO (Hierarchy of Input, Processing, and Output) Charts

5.4.0 Read and Write Interface Documents for Hardware, Software, and Data

6.0 ALGORITHM DEVELOPMENT

6.1.0 Read and Write a Top-Down Design

6.2.0 Write Stepwise Refinement

6.3.0 Create and Use a Modular Design

6.4.0 Use Meaningful Variables

6.5.0 Select Appropriate Data Structures

6.6.0 Apply Memory Management Techniques

6.7.0 Use Number Systems

7.0 PROGRAM CODE PRODUCTION

7.1.0 Write Program Comments

7.2.0 Construct Sub-Programs

7.3.0 Use Selection (Branching) Methods

7.4.0 Use Iteration (Looping) Methods

7.5.0 Write I/O (Input/Output) Commands

7.6.0 Write System Calls

7.7.0 Convert Data Among Number Systems

7.8.0 Represent Data in Various Data Types

7.9.0 Write Code to a Specified Style

7.10.0 Use Simple Data Types

7.11.0 Use Arrays

7.12.0 Use Records

7.13.0 Use Files

7.14.0 Use Pointers

7.15.0 Use Linked Lists

7.16.0 Use Trees

8.0 PROGRAM CODE TESTING

8.1.0 Identify and Test Critical Points

8.2.0 Test All Branch Possibilities

8.3.0 Test for Invalid Input

8.4.0 Test for Logical Correctness

8.5.0 Test Output Format

8.6.0 Test for Functional Correctness and Completeness

8.7.0 Apply Intermediate Testing Techniques

8.8.0 Identify Unusual Program Behavior

9.0 PROGRAM CODE MAINTENANCE

9.1.0 Read and Analyze Code

9.2.0 Modify Program Behaviors

9.3.0 Implement Formal Specification Changes

10.0 HARDWARE DIAGNOSIS AND TROUBLE-SHOOTING

10.1.0 Run Hardware Diagnostic Programs

10.2.0 Use Basic External and Internal Hardware Concepts

10.3.0 Use Basic Architecture Concepts

11.0 ORAL COMMUNICATION SKILLS

11.1.0 Give a Program Design Presentation

11.2.0 Use Active Listening Skills

11.3.0 Use Good Questioning Techniques

11.4.0 Defend an Idea

11.5.0 Use Public Speaking Skills

11.6.0 Use Good Telephone Skills

12.0 WRITTEN COMMUNICATION SKILLS

12.1.0 Write Memoranda

12.2.0 Write Business Letters

12.3.0 Write Reports

12.4.0 Write User Documentation to Specified Format

12.4.1 Write a Software Requirement Document (SRD)

12.4.2 Rough-Out a Program Reference Manual

12.4.3 Rough-Out a User Guide

13.0 HUMAN RELATIONS SKILLS

13.1.0 Demonstrate Punctuality

13.2.0 Schedule and Use Time Effectively

13.3.0 Exhibit Appropriate Dress and Grooming

13.4.0 Exhibit Ethical Behavior

13.5.0 Demonstrate the Ability to Compromise

13.6.0 Demonstrate Leadership Techniques

13.7.0 Demonstrate Knowledge of Basic Social Graces

13.8.0 Use Tact

## DACUM EXPLANATION

The following pages are the validation forms sent to various companies for some of their workers to fill out.

# PART I - TASK STATEMENTS

**Instructions: Please read very carefully!**

On the pages which follow you will find a list of task statements clustered into thirteen major duties (1 through 13) which may relate to your software development job. We need your personal reaction to each task statement:

How important is the performance of the task in your job as a software developer?

Answer this question by completing the following steps:

1. For each task indicate how important you believe performance of the task is in your job as a software developer. In the "Importance of Task" column choose and circle the number which most accurately reflects the importance of that task. If you do not perform some of the tasks because they are another software developer's responsibility, please circle the number which indicates how important you believe those tasks are to the overall job performance of your company. Use the scale below to rate the importance of each task:

| | |
|---|---|
| 5 = Of Great Importance | Performance is critical to program operation. |
| 4 | |
| 3 | |
| 2 | |
| 1 | |
| 0 = Of No Importance | Performance makes no contribution to program operation. |

2. Add any statements that describe other tasks that you have performed or that you feel need to be performed by software developers which are not listed.

# PART II. GENERAL INFORMATION

**Instructions: Read each item carefully and write, or check where appropriate, your responses in the blanks provided.**

1. Indicate your present job title:_____

2. Indicate which of the following best describes your education:
   - ( ) Less than high school diploma
   - ( ) High school diploma of GED
   - ( ) Two-year associate degree
   - ( ) Bachelor's degree
   - ( ) Master's degree
   - ( ) PhD degree

3. Please indicate whether you are employed by a company of:
   - _____a. Less than 50 employees.
   - _____b. Less than 500 employees.
   - _____c. Greater than 500 employees.

4. What percentage of your time is devoted to software development? _____%

5. Indicate the total number of years experience that you have had as a software developer: _____years.

*THANK YOU VERY MUCH FOR COMPLETING THIS INVENTORY, PLEASE PLACE IT IN THE ENCLOSED ENVELOPE AND MAIL TO COLLIN COUNTY COMMUNITY COLLEGE.*

# SOFTWARE DEVELOPMENT TASK INVENTORY

## Importance of Task

**How Important is the performance of this task in your job as a software developer?**

(Circle one response)

| | | Great Importance | | | | | No Importance |
|---|---|---|---|---|---|---|---|
| **DUTY 1: BASIC COMPUTER SKILLS** | | | | | | | |
| 1. | Identify and Use Computer Terminology | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Boot and/or Logon to a Computer | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use Operating System Commands | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Manage a File | 5 | 4 | 3 | 2 | 1 | 0 |
| | 1. Name a File | 5 | 4 | 3 | 2 | 1 | 0 |
| | 2. Create and Use Subdirectories | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| **DUTY 2: COMPUTER APPLICATIONS PROGRAMS** | | | | | | | |
| 1. | Use a Word Processing Package | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Use a Database Manager | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use an Electronic Spreadsheet | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Use a Terminal Emulator | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 3: SOFTWARE DEVELOPMENT TOOLS**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Use a Text Editor | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Use an Interpreter | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use a Compiler | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Use an Assembler | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Use a Linker | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | Use a Loader | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | Use a Symbolic Debugger | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | Use Assembly Language | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | Use at Least One High-Level Language | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 11. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 12. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 13. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 4: DOCUMENTATION**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Follow Documentation Instructions | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Use Documentation to Learn New Software | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Apply Cross-Referencing Techniques | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 5: SYSTEM DESIGN**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Read and Write Pseudo Code | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Read and Write Flow Charts | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Read and Write HIPO (Hierarchy of Input, Processing, and Output) Charts | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4. | Read and Write Interface Documents for Hardware, Software and Data | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 6: ALGORITHM DEVELOPMENT**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Read and Write a Top-Down Design | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Write Stepwise Refinement | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Create and Use a Modular Design | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Use Meaningful Variables | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Select Appropriate Data Structures | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | Apply Memory Management Techniques | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | Use Number Systems | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 11. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 12. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 7: PROGRAM CODE PRODUCTION**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Write Program Comments | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Construct Sub-Programs | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use Selection (Branching) Method | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Use Iteration (Looping) Methods | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Write I/O (Input/Output) Commands | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | Write System Calls | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7. | Convert Data Among Number Systems | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | Represent Data in Various Data Types | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | Write Code to a Specified Style | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | Use Simple Data Types | 5 | 4 | 3 | 2 | 1 | 0 |
| 11. | Use Arrays | 5 | 4 | 3 | 2 | 1 | 0 |
| 12. | Use Records | 5 | 4 | 3 | 2 | 1 | 0 |
| 13. | Use Files | 5 | 4 | 3 | 2 | 1 | 0 |
| 14. | Use Pointers | 5 | 4 | 3 | 2 | 1 | 0 |
| 15. | Use Linked Lists | 5 | 4 | 3 | 2 | 1 | 0 |
| 16. | Use Trees | 5 | 4 | 3 | 2 | 1 | 0 |
| 17. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 18. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 19. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 20. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 21. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 22. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 23. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 24. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 25. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 8: PROGRAM CODE TESTING**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Identify and Test Critical Points | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Test All Branch Possibilities | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Test for Invalid Input | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Test for Logical Correctness | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Test Output Format | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | Test for Functional Correctness and Completeness | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7. | Apply Intermediate Testing Techniques | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | Identify Unusual Program Behavior | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 11. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 12. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 9: PROGRAM CODE MAINTENANCE**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Read and Analyze Code | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Modify Program Behaviors | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Implement Formal Specification Changes | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 10: HARDWARE DIAGNOSIS AND TROUBLE-SHOOTING**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Run Hardware Diagnostic Programs | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Use Basic External and Internal Hardware Concepts | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use Basic Architecture Concepts | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 11: ORAL COMMUNICATION SKILLS**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Give a Program Design Presentation | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Use Active Listening Skills | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Use Good Questioning Techniques | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4. | Defend an Idea | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Use Public Speaking Skills | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | Use Good Telephone Skills | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 12: WRITTEN COMMUNICATION SKILLS**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Write Memoranda | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Write Business Letters | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Write Reports | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Write User Documentation to Specified Format | 5 | 4 | 3 | 2 | 1 | 0 |
| | 1. Write a Software Requirements Document (SRD) | 5 | 4 | 3 | 2 | 1 | 0 |
| | 2. Rough-Out a Program Reference Manual | 5 | 4 | 3 | 2 | 1 | 0 |
| | 3. Rough-Out a User Guide | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 6. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

**DUTY 13: HUMAN RELATIONS SKILLS**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | Demonstrate Punctuality | 5 | 4 | 3 | 2 | 1 | 0 |
| 2. | Schedule and Use Time Effectively | 5 | 4 | 3 | 2 | 1 | 0 |
| 3. | Exhibit Appropriate Dress and Grooming | 5 | 4 | 3 | 2 | 1 | 0 |
| 4. | Exhibit Ethical Behavior | 5 | 4 | 3 | 2 | 1 | 0 |
| 5. | Demonstrate the Ability to Compromise | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6. | Demonstrate Leadership Techniques | 5 | 4 | 3 | 2 | 1 | 0 |
| 7. | Demonstrate Knowledge of Basic Social Graces | 5 | 4 | 3 | 2 | 1 | 0 |
| 8. | Use Tact | 5 | 4 | 3 | 2 | 1 | 0 |
| 9. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 10. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 11. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 12. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 13. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 14. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |
| 15. | _____ | 5 | 4 | 3 | 2 | 1 | 0 |

COMMENTS:

## DACUM EXPLANATION

The following pages are the validated DACUM charts, compiled from the competencies acquired from the DACUM panel and the validation forms returned from industry.

| DUTIES | TASKS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1.0 BASIC COMPUTER SKILLS | 1.1.0 Identify and Use Computer Terminology | 1.2.0 Boot and/or Logon to a Computer | 1.3.0 Use Operating System Commands | 1.4.0 Manage a File | 1.4.1 Name a File | 1.4.2 Create and Use Subdirectories | 1.5.0 Create Command Files | 1.6.0 Back-up & Restore Hard Drives | 1.7.0 Format Disks |
| 2.0 COMPUTER APPLICATIONS PROGRAMS | 2.1.0 Use a Word Processing Package | 2.2.0 Use A Database Manager | 2.3.0 Use an Electronic Spreadsheet | 2.4.0 Use a Terminal Emulator | | | | | |
| 3.0 SOFTWARE DEVELOPMENT TOOLS | 3.1.0 Use a Text Editor | 3.2.0 Use an Interpreter | 3.3.0 Use a Compiler | 3.4.0 Use an Assembler | 3.5.0 Use a Linker | 3.6.0 Use a Loader | 3.7.0 Use a Symbolic Debugger | 3.8.0 Use Assembly Language | 3.9.0 Use at Least One High-Level Language |
| 4.0 DOCUMENTATION | 4.1.0 Follow Documentation Instructions | 4.2.0 Use Documentation to Learn New Software | 4.3.0 Apply Cross-Referencing Techniques | | | | | | |
| 5.0 SYSTEM DESIGN | 5.1.0 Read and Write Pseudo Code | 5.2.0 Read and Write Flow Charts | 5.3.0 Read and Write HIPO (Hierarchu of Input, Processing, and Output) | 5.4.0 Read and Write Interface Documents for Hardware, Software, | 5.5.0 Select Appropriate System Architecture | 5.6.0 Select Appropriate Design Notation | 5.7.0 Read & Write Data Flow Diagrams | 5.8.0 Read & Write Structure Charts | |
| 6.0 ALGORITHM DEVELOPMENT | 6.1.0 Read and Write a Top-Down Design | 6.2.0 Write Stepwise Refinement | 6.3.0 Create and Use a Modular Design | 6.4.0 Use Meaningful Variables | 6.5.0 Select Appropriate Data Structures | 6.6.0 Apply Memory Management Techniques | 6.7.0 Use Number Systems | 6.8.0 Adapt to Styles of Others | 6.9.0 Interpret Designs of Others |
| 7.0 PROGRAM CODE PRODUCTION | 7.1.0 Write Program Comments | 7.2.0 Construct Sub-Programs | 7.3.0 Use Selection (Branching) Methods | 7.4.0 Use Iteration (Looping) Metheds | 7.5.0 Write I/O (input/Output) Commands | 7.6.0 Write System Calls | 7.7.0 Convert Data Among Number Systems | 7.8.0 Represent Data in Various Data Types | 7.9.0 Write Code to a Specified Style |

| DUTIES | TASKS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7.0 PROGRAM CODE PRODUCTION | 7.10.0 Use Simple Data Types | 7.11.0 Use Arrays | 7.12.0 Use Records | 7.13.0 Use Files | 7.14.0 Use Pointers | 7.15.0 Use Linked Lists | 7.16.0 Use Trees | 7.17.0 Constructively Critique Code from Others | |
| 8.0 PROGRAM CODE TESTING | 8.1.0 Identify and Test Critical Points | 8.2.0 Test All Branch Possibilities | 8.3.0 Test for Invalid Input | 8.4.0 Test for Logical Correctness | 8.5.0 Test Output Format | 8.6.0 Test for Functional Correctness and Completeness | 8.7.0 Apply Intermediate Testing Techniques | 8.8.0 Identify Unusual Program Behavior | 8.9.0 Clearly Document Unusual Behavior |
| 9.0 PROGRAM CODE MAINTENANCE | 9.1.0 Read and Analyze Code | 9.2.0 Modify Program Behaviors | 9.3.0 Implement Formal Specification Changes | 9.4.0 Interpret Formal Specification Changes | | | | | |
| 10.0 HDWARE DIAG. & TRBL-SHOOTING | 10.1.0 Run Hardware Diagnostic Programs | 10.2.0 Use Basic External and Internal Hardware Concepts | 10.3.0 Use Basic Architecture Concepts | | | | | | |
| 11.0 ORAL COMMUNICATION SKILLS | 11.1.0 Give a Program Design Presentation | 11.2.0 Use Active Listening Skills | 11.3.0 Use Good Questioning Techniques | 11.4.0 Defend an Idea | 11.5.0 Use Public Speaking Skills | 11.6.0 Use Good Telephone Skills | | | |
| 12.0 WRITTEN COMMUNICATION SKILLS | 12.1.0 Write Memoranda | 12.2.0 Write Business Letters | 12.3.0 Write Reports | 12.4.0 Write User Documentation to Specified Format | 12.4.1.0 Write a Software Requirements Document (SRD) | 12.4.2.0 Rough-Out a Program Reference Manual | 12.4.3.0 Rough-Out a User Guide | 12.5.0 Write Test Plans | |
| 13.0 HUMAN RELATIONS SKILLS | 13.1.0 Demonstrate Punctuality | 13.2.0 Schedule and Use Time Effectively | 13.3.0 Exhibit Appropriate Dress and Grooming | 13.4.0 Exhibit Ethical Behavior | 13.5.0 Demonstrate the Ability to Compromise | 13.6.0 Demonstrate Leadership Techniques | 13.7.0 Demonstrate Knowledge of Basic Social Graces | 13.8.0 Use Tact | 13.9.0 Function as Effective Team Member |

**DUTIES**

**TASKS**

| 13.0 HUMAN RELATIONS SKILLS | 13.10.0 Understand importance of Quality | 13.11.0 Demonstrate Dependability | 13.12.0 Handle Difficult People. | 13.13.0 Accept Criticism | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

# CHAPTER THREE

# CHAPTER 3: DEGREE PLANS, COURSE DESCRIPTIONS, AND COURSE COMPETENCIES

1. Degree plans.
    1.1. Page 3-55
    1.2. Page 3-56
    1.3. Page 3-57
2. Catalog course descriptions Page 3-58
3. Course learning objectives and descriptions.
    3.1. Computer Organization and Assembly Language Programming.
    3.2. Programming Concepts I.
    3.3. Programming Concepts II.
    3.4. Software Engineering.
    3.5. Intro to Artificial Intelligence.
    3.6. Scientific Programming.
    3.7. C Programming Language.
    3.8. Real Time Programming
    3.9. Software Test Techniques
    3.10. Computer Graphics systems
    3.11. Advanced Software Engineering
    3.12. Advanced Assembly Language Programming
    3.13. Networking & Telecommunications
    3.14. LISP Programming
    3.15. Basics of Electronics
    3.16. Composition/Rhetoric I
    3.17. College Algebra
    3.18. Applied Psychology
    3.19. Fundamentals of Speech Communication

3.20. Introduction to Economics

3.21. Introduction to Humanities

3.22. Trigonometry

3.23. Technical Writing

## COLLIN COUNTY COMMUNITY COLLEGE DISTRICT

# ASSOCIATE OF APPLIED SCIENCE IN ENGINEERING TECHNOLOGY

## MAJOR: SOFTWARE DEVELOPMENT

| | | | CR HRS | SEM COMPL | GRADE RECVD |
|---|---|---|---|---|---|
| **I. GENERAL EDUCATION CORE: (22 credit hours)** | | | | | |
| A. | ENGL 151 | Composition/Rhetoric I | 3 | ____ | ____ |
| B. | MATH 181 | College Algebra | 3 | ____ | ____ |
| C. | CPSC 190 | Programming Concepts I | 3 | ____ | ____ |
| D. | PSYC 121 | Applied Psychology | 3 | ____ | ____ |
| E. | SPCM 151 | Fund. of Speech Comm | 3 | ____ | ____ |
| F. | ECON 121 | Intro to Economics | 3 | ____ | ____ |
| G. | HUM 151 | Intro to Humanities | 3 | ____ | ____ |
| H. | HPER | Elective | 1 | ____ | ____ |
| **II. TECHNICAL PROGRAM CORE: (9 credit hours)** | | | | | |
| A. | ELEC 131 | Basic Electronics | 3 | ____ | ____ |
| B. | MATH 182 | Trigonometry | 3 | ____ | ____ |
| C. | ENGL 291 | Technical Writing | 3 | ____ | ____ |
| **III. MAJOR COURSE: (28 credit hours)** | | | | | |
| A. | CPSC 191 | Programming Concepts II | 3 | ____ | ____ |
| B. | CSFT 121 | Comp. Org. & Assembly Lang | 4 | ____ | ____ |
| C. | CSFT 225 | Ada Programming | 3 | ____ | ____ |
| D. | CPSC 294 | C Programming | 3 | ____ | ____ |
| E. | CPSC 292 | Scientific Programming | 3 | ____ | ____ |
| F. | CSFT 221 | Software Engineering | 3 | ____ | ____ |
| G. | CSFT 223 | Real Time Programming | 3 | ____ | ____ |
| H. | CSFT 224 | Software Test Techniques | 3 | ____ | ____ |
| I. | CSFT 232 | Advanced Software Engr. | 3 | ____ | ____ |
| **IV. ELECTIVES: (6 credit hours)** | | | | | |
| A. | CSFT 233 | Advanced Assembly Lang Prog | 3 | ____ | ____ |
| B. | CIS 262 | Computer Graphics Systems | 3 | ____ | ____ |
| C. | CSFT 236 | Intro to Artificial Intell | 3 | ____ | ____ |
| D. | CSFT 235 | LISP Programming | 3 | ____ | ____ |
| E. | CIS 270 | Networking & Telecomm. | 3 | ____ | ____ |

# ASSOCIATE OF APPLIED SCIENCE IN ENGINEERING TECHNOLOGY

## MAJOR: SOFTWARE DEVELOPMENT

| | | | LEC. HRS. WK. | LAB HRS. WK. | CONT. HRS. SEM. | CREDIT HRS. SEM. |
|---|---|---|---|---|---|---|
| **SEMESTER I:** | | | | | | |
| CPSC | 190 | Programming Concepts I | 3 | 2 | 80 | 3 |
| CPSC | 191 | Programming Concepts II | 3 | 2 | 80 | 3 |
| MATH | 181 | College Algebra | 3 | 0 | 48 | 3 |
| ENGL | 151 | Composition/Rhetoric | 3 | 1 | 64 | 3 |
| ECON | 121 | Introduction to Economics | 3 | 1 | 64 | 3 |
| | | | | | 336 | 15 |
| **SEMESTER II:** | | | | | | |
| CSFT | 121 | Comp. Org. & Assembly Language | 4 | 2 | 96 | 4 |
| MATH | 182 | Trigonometry | 3 | 0 | 48 | 3 |
| ELEC | 131 | Basic Electronics | 2 | 2 | 64 | 3 |
| SPCM | 124 | Fundamentals of Speech | 3 | 1 | 64 | 3 |
| HPER | | Physical Activity | 1 | 2 | 48 | 1 |
| | | | | | 320 | 14 |
| **SUMMER:** | | | | | | |
| CSFT | 225 | Ada Programming | 3 | 2 | 80 | 3 |
| CPSC | 294 | C Programming | 3 | 2 | 80 | 3 |
| | | | | | 160 | 6 |
| **SEMESTER III:** | | | | | | |
| HUM | 151 | Intro to Humanities | 3 | 1 | 64 | 3 |
| CPSC | 292 | Scientific Programming | 3 | 2 | 80 | 3 |
| CSFT | 221 | Software Engineering | 3 | 2 | 80 | 3 |
| ENGL | 291 | Technical Writing | 3 | 1 | 64 | 3 |
| CSFT | | Elective | 3 | 2 | 80 | 3 |
| | | | | | 368 | 15 |
| **SEMESTER IV:** | | | | | | |
| CSFT | 223 | Real Time Programming | 3 | 2 | 80 | 3 |
| CSFT | 224 | Software Test Techniques | 3 | 2 | 80 | 3 |
| CSFT | | Elective | 3 | 2 | 80 | 3 |
| PSYC | 121 | Applied Psychology | 3 | I | 64 | 3 |
| CSFT | 232 | Advanced Software Engineering | 3 | 2 | 80 | 3 |
| | | | | | 384 | 15 |
| | | **TOTAL** | | | **1568** | **65** |

# ASSOCIATE OF APPLIED SCIENCE

## WORKSHEET FOR EMPHASIS IN: SOFTWARE DEVELOPMENT

**********

**SUGGESTED SEMESTER PLAN FOR 2 YEAR COMPLETION**

**********

**SEMESTER I**

| | |
|---|---|
| CPSC 190 | Programming Concepts I |
| CPSC 191 | Programming Concepts II |
| MATH 181 | College Algebra |
| ENGL 151 | Composition/Rhetoric |
| ECON 121 | Introduction to Economics |

**SEMESTER II**

| | |
|---|---|
| CSFT 121 | Comp Org & Assembly Language |
| MATH 182 | Trigonometry |
| ELEC 131 | Basic Electronics |
| SPCM 151 | Fundamentals of Speech |
| HPER | Physical Activity |

**SEMESTER III**

| | |
|---|---|
| HUM 151 | Intro to Humanities |
| CPSC 292 | Scientific Programming |
| CSFT 221 | Software Engineering |
| ENGL 291 | Technical Writing |
| CSFT | Elective |

**SEMESTER IV**

| | |
|---|---|
| CSFT 223 | Real Time Programming |
| CSFT 224 | Software Test Techniques |
| CSFT | Elective |
| PSYC 121 | Applied Psychology |
| CSFT 232 | Advanced Software Engr. |

**SUMMER**

| | |
|---|---|
| CSFT 225 | Ada Programming |
| CPSC 294 | C Programming |

**********

**SUGGESTED SEMESTER PLAN FOR 4 YEAR COMPLETION**

**********

**SEMESTER I**

| | |
|---|---|
| CPSC 190 | Programming Concepts I |
| CPSC 191 | Programming Concepts II |
| MATH 181 | College Algebra |

**SEMESTER II**

| | |
|---|---|
| ELEC 131 | Basic Electronics |
| CSFT 121 | Comp Org & Assembly Language |

**SEMESTER III**

| | |
|---|---|
| ENGL 151 | Composition/Rhetoric |
| MATH 182 | Trigonometry |
| CSFT 225 | Ada Programming |

**SEMESTER IV**

| | |
|---|---|
| CPSC 294 | C Programming |
| ECON 121 | Introduction to Economics |

**SEMESTER V**

| | |
|---|---|
| CPSC 292 | Scientific Programming |
| CSFT 221 | Software Engineering |
| ENGL 291 | Technical Writing |

**SEMESTER VI**

| | |
|---|---|
| CSFT 224 | Software Test Techniques |
| CSFT 223 | Real Time Programming |
| HPER | Physical Activity |

**SEMESTER VII**

| | |
|---|---|
| PSYC 121 | Applied Psychology |
| CSFT | Elective |
| HUM 151 | Intro to Humanities |

**SEMESTER VIII**

| | |
|---|---|
| SPCM 151 | Fundamentals of Speech |
| CSFT 232 | Advanced Software Engr. |
| CSFT | Elective |

# CATALOG COURSE DESCRIPTIONS

**CPSC 190 PROGRAMMING CONCEPTS I**

A study of logical operation and organization of a computer, number systems, boolean algebra, program solving techniques, algorithmic processes, and top-down design using the PASCAL language. COREQUISITE: MATH 181 or consent of instructor. Lab required. 3 credit hours.

**CPSC 191 PROGRAMMING CONCEPTS II**

Continuation of Computer Science 190, including structured programming, design, data structures, documentation, and file processing. Emphasis on creating and modifying larger programs. PREREQUISITE: CPSC 190 or consent of instructor. Lab required. 3 credit hours.

**CSFT 121 ASSEMBLY LANGUAGE PROGRAMMING**

Low level programming techniques using assembly language. Algorithm development, operation codes, addressing, pointers, looping, control structures, top-down design, structured programming, macros, subroutines, argument passing, debugging techniques, local and global variables, recursion, stacks, registers, timing, and number systems. PREREQUISITE: CPSC 191 or consent of instructor. Lab required. 4 credit hours.

**CSFT 225 ADA PROGRAMMING**

Syntax and semantics of ADA language, packages, I/O, encapsulation, tasking, blocks, exceptions, private and generic types. PREREQUISITE: CSFT 121 or CPSC 290 or consent of instructor. Lab required. 3 credit hours.

**CPSC 294 C PROGRAMMING**

Study of C programming language with emphasis on structured approach to program design and documentation. Topics include looping, arrays, functions, structures, and bit operations. Scientific, business, and computer arithmetic applications. PREREQUISITE: CFST 121 or CPSC 290 or consent of Instructor. Lab required. 3 credit hours.

**CPSC 292 SCIENTIFIC PROGRAMMING**

Introduction to numerical techniques with applications in science and engineering using FORTRAN. Emphasis on program design and documentation. Topics include subscripting, file processing, and subroutines. PREREQUISITE: MATH 181 and CSFT 121 or CPSC 290 or consent of instructor. Lab required. 3 credit hours.

**CSFT 221 SOFTWARE ENGINEERING**

The study of software design, implementation, and validation techniques through team projects. Structured analysis, programming style and project documentation are emphasized in software projects large enough to give a group meaningful work experience. PREREQUISITE: CSFT 121 or CPSC 290 or consent of instructor. Lab required. 3 credit hours.

**CSFT 223 REAL TIME PROGRAMMING**

Analysis of distributed networks containing mini and micro computers. Study of data acquisition and digital control environments. PREREQUISITE: CPSC 292 and MATH 182 or consent of instructor. Lab required. 3 credit hours.

**CSFT 224 SOFTWARE TEST TECHNIQUES**

Introduction to software testing methodologies. Emphasis on program development techniques which aid testing. Introduction to proof of correctness. Laboratory exercises assigned to reinforce principles of program development. PREREQUISITE: CSFT 121 or CPSC 290 and MATH 182 or consent of instructor. Lab required. 3 credit hours.

**CIS 262 COMPUTER GRAPHICS SYSTEMS**

Concepts and principles, survey of preset display and input techniques, systems and applications. Study of basic concepts, mathematical and geometric principles, design and use of graphics software packages, design and implementation of an application using available hardware and software. PREREQUISITE: CSFT 121 or CPSC 290 and Math 182 or consent of instructor. Lab required. 3 credit hours.

**CSFT 232 ADVANCED SOFTWARE ENGINEERING**

Advanced study of large program design and documentation, group problems, software maintenance and reliability. PREREQUISITE: CSFT 221 and Math 182 or consent of instructor. Lab required. 3 credit hours.

**CSFT 233 ADVANCED ASSEMBLY LANGUAGE PROGRAMMING**

Program design and practice with assembly languages, macro definitions, conditioned assembly, advanced I/O, floating point operations. PREREQUISITE: CSFT 121 or CPSC 290 and MATH 182 or consent of instructor. Lab required. 3 credit hours.

**CIS 270 NETWORKING & TELECOMMUNICATIONS**

This course reviews data, text, graphics, and voice communications technology and their applications. Included is vocabulary, configuration of local networks, modems, rates, and standards. An overview of protocols is given. PREREQUISITE: CPSC 150, CIS 210. Lab required. 3 credit hours.

**CSFT 235 LISP PROGRAMMING**

Syntax and semantics of LISP programming language, style, recursion, tail recursion, algorithm development, list processing techniques. PREREQUISITE: CSFT 121 or CPSC 290 and MATH 182 or consent of instructor. Lab required. 3 credit hours.

**CSFT 236 INTRO TO ARTIFICIAL INTELLIGENCE**

Introduction to concepts and ideas in artificial intelligence. Topics will include search techniques, knowledge representation, control strategies and advanced problem-solving architecture. PREREQUISITE: CSFT 235 or consent of instructor. Lab required. 3 credit hours.

# COURSE OUTLINE EXPLANATION

The following pages contain the course descriptions and learning objectives of the Software Development Program. The learning objectives are the validated competencies matched with courses. Many appear in more than one course. The first time a competency appears it is indicated with "(I)". The subsequent appearences with "(X)", indicating reinforcement of the competency.

# CSFT 121 COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

## COURSE DESCRIPTION

Enough assembly language details should be covered and projects assigned so that the student gains experience in programming a specific computer. However, concepts and techniques that apply to a broad range of computers should be emphasized. Programming methods that are developed in Programming Concepts I and Programming Concepts II should also be utilized in this course.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Boot and/or log on to a computer
(X) Use operating system commands
(X) Manage a file
(X) Name a file
(X) Create and use subdirectories
(X) Create command files
(X) Backup & restore hard drives
(X) Format disks
(I) Use a terminal emulator
(X) Use a text editor
(X) Use an interpreter
(X) Use a compiler
(I) Use an assembler
(I) Use a linker
(I) Use a symbolic debugger
(I) Use assembly language
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Use documentation to learn new software
(I) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(I) Select appropriate system architecture
(I) Select appropriate design notation
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Use meaningful variables
(X) Select appropriate data structures
(X) Use number systems
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write program comments
(X) Construct subprograms
(X) Use selection (branching) methods
(X) Use iteration (looping) methods
(X) Write I/O (Input/Output) commands
(X) Write system calls
(I) Convert data among number systems
(X) Represent data in various data types

(X) Write code to a specified style
(X) Use simple data types
(X) Use arrays
(X) Use records
(X) Use files
(X) Use pointers
(X) Use linked lists
(X) Use trees
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(I) Identify unusual program behavior
(X) Read and analyze code
(X) Modify program behaviors
(I) Use basic external and internal hardware concepts
(I) Use basic architecture concepts
(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(X) Demonstrate the ability to compromise
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Handle difficult people
(X) Accept criticism

# CPSC 190 PROGRAMMING CONCEPTS I

## COURSE DESCRIPTION

The material on a high level programming language and on algorithm development can be taught best as an integrated whole. Thus the topics should not be covered sequentially. The emphasis of the course is on the techniques of algorithm development and programming with style. Neither esoteric features of a programming language nor other aspects of computers should be allowed to interfere with that purpose.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(I) Identify and use computer terminology
(I) Boot and/or log on to a computer
(I) Use operating system commands
(I) Manage a file
(I) Name a file
(I) Format disks
(I) Use a text editor
(I) Use an interpreter
(I) Use a compiler
(I) Use at least one high-level language
(I) Follow documentation instructions
(I) Use documentation to learn new software
(I) Read and write pseudo code
(I) Read and write flow charts
(I) Read and write a top-down design
(I) Write stepwise refinement
(I) Use meaningful variables
(I) Use number systems
(I) Write program comments
(I) Construct subprograms
(I) Use selection (branching) methods
(I) Use iteration (looping) methods
(I) Write I/O (Input/Output) commands
(I) Represent data in various data types
(I) Write code to a specified style
(I) Use simple data types
(I) Use arrays
(I) Use files
(I) Identify and test critical points
(I) Test all branch possibilities
(I) Test for invalid input
(I) Test for logical correctness
(I) Read and analyze code
(I) Modify program behaviors
(I) Demonstrate punctuality
(I) Schedule and use time effectively
(I) Understand importance of quality
(I) Accept criticism

*Learning objectives are specifically for Software Development, not a complete list.

# CPSC 191 PROGRAMMING CONCEPTS II

## COURSE DESCRIPTION

The topics in this outline should be introduced as needed in the context of one or more projects involving large programs. The instructor may choose to begin with the statement of a sizeable project, then utilize structured programming techniques to develop a number of small projects each of which involves string processing, recursion, searching and sorting, or data structures. The emphasis on good programming style, expression, and documentation, begun in Programming Concepts I, should be continued. Analysis of algorithms should be introduced, but at this level such analysis should be given by the instructor to the student.

Consideration should be given to the implementation of programming projects by organizing students into programming teams. This technique is essential in advanced level courses and should be attempted as early as possible in the curriculum. If large class size makes such an approach impractical, every effort should be made to have each student's programs read and critiqued by another student.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Boot and/or log on to a computer
(X) Use operating system commands
(X) Manage a file
(X) Name a file
(I) Create and use subdirectories
(I) Create command files
(I) Backup & restore hard drives
(X) Format disks
(X) Use a text editor
(X) Use an interpreter
(X) Use a compiler
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Use documentation to learn new software
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Use meaningful variables
(I) Select appropriate data structures
(X) Use number systems
(I) Adapt to styles of others
(I) Interpret designs of others
(X) Write program comments
(X) Construct subprograms
(X) Use selection (branching) methods
(X) Use iteration (looping) methods
(X) Write I/O (Input/Output) commands
(I) Write system calls
(X) Represent data in various data types
(X) Write code to a specified style
(X) Use simple data types

(X) Use arrays
(I) Use records
(X) Use files
(I) Use pointers
(I) Use linked lists
(I) Use trees
(I) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Read and analyze code
(X) Modify program behaviors
(I) Use active listening skills
(I) Use good questioning techniques
(I) Defend an idea
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(I) Demonstrate the ability to compromise
(I) Demonstrate leadership techniques
(I) Use tact
(I) Function as effective team member
(X) Understand importance of quality
(I) Demonstrate dependability
(I) Handle difficult people
(X) Accept criticism

*Learning objectives are specifically for Software Development, not a complete list.

# CSFT 221 SOFTWARE ENGINEERING

## COURSE DESCRIPTION

This course presents a formal approach to state-of-the-art techniques in software design and development and provides a means for students to apply the techniques. An integral part of the course is the involvement of students working in teams in the organization, management, and development of a large software project. The team project aspect can be facilitated either by scheduling separate laboratories or by using some of the lecture periods to discuss practical aspects of the team projects.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Follow documentation instructions
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(I) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(I) Read and write interface documents for hardware, software and data
(X) Select appropriate system architecture
(X) Select appropriate design notation
(I) Read and write data flow diagrams
(I) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Create and use a modular design
(X) Select appropriate data structures
(I) Apply memory management techniques
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write program comments
(X) Write system calls
(X) Represent data in various data types
(X) Write code to a specified style
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Identify unusual program behavior
(I) Clearly document unusual behavior
(X) Read and analyze code
(X) Modify program behaviors
(I) Implement formal specification changes
(I) Interpret formal specification changes
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(I) Give a program design presentation
(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(I) Write user documentation to specified format
(I) Write a software requirements document(SRD)

(I) Rough-out a program reference manual
(I) Rough-out a user guide
(X) Schedule and use time effectively
(X) Demonstrate the ability to compromise
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Handle difficult people
(X) Accept criticism

# CSFT 236 INTRO TO ARTIFICIAL INTELLIGENCE

## COURSE DESCRIPTION

This course introduces students to basic concepts and techniques of artificial intelligence, or intelligent systems, and gives insights into active research areas and applications. Emphasis is placed on representation as a central and necessary concept for work in intelligent systems. Strategies for choosing representations as well as notational systems and structures should be discussed. Students should understand, for example, that the selection of a programming language is really a basic representational choice and that an important component of that choice is whether the programming language is really the basic representational mode or whether it is a translator/interpreter of an intermediate representational mode such as the predicate calculus or other notational system (e.g. modal or fuzzy logics).

Other issues of importance in this course are natural language, vision systems, search strategies, and control. The extent and type of coverage will vary. The use of natural language and vision systems in applications of intelligent systems research to other disciplines should be emphasized. Search strategies should be seen as being implicit in representation and control. General issues related to control should be discussed and illustrated by examples of existing systems. A variety of applications could be mentioned at the beginning of the course as motivation for studying intelligent systems. These applications could then be elaborated on at appropriate times throughout the course or at the end.

Students could profit from a background in LISP because of its widespread use in artificial intelligence work.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Use a text editor
(X) Use an interpreter
(X) Use a compiler
(X) Use a symbolic debugger
(X) Use at least one high-level language
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(X) Read and write interface documents for hardware, software and data
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Create and use a modular design
(X) Apply memory management techniques
(X) Write program comments
(X) Write system calls
(X) Represent data in various data types
(X) Identify and test critical points
(X) Test for invalid input
(X) Test for logical correctness
(X) Test for functional correctness and completeness
(X) Apply intermediate testing techniques

(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Write test plans
(X) Schedule and use time effectively

# CPSC 292 SCIENTIFIC PROGRAMMING

## COURSE DESCRIPTION

Numerical techniques and FORTRAN-77 are probably new to the student, but the continuation of problem solving methods, algorithm development techniques, program design, coding style, debugging tools, and proper program documentation is to be concentrated upon also.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Boot and/or log on to a computer
(X) Use operating system commands
(X) Manage a file
(X) Name a file
(X) Create and use subdirectories
(X) Create command files
(X) Backup & restore hard drives
(X) Format disks
(X) Use a terminal emulator
(X) Use a text editor
(X) Use a compiler
(X) Use an assembler
(X) Use a linker
(X) Use a symbolic debugger
(X) Use assembly language
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Use documentation to learn new software
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Select appropriate system architecture
(X) Select appropriate design notation
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Use meaningful variables
(X) Select appropriate data structures
(X) Use number systems
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write program comments
(X) Construct subprograms
(X) Use selection (branching) methods
(X) Use iteration (looping) methods
(X) Write I/O (Input/Output) commands
(X) Write system calls
(X) Convert data among number systems
(X) Represent data in various data types
(X) Write code to a specified style
(X) Use simple data types
(X) Use arrays
(X) Use records

(X) Use files
(X) Use pointers
(X) Use linked lists
(X) Use trees
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Identify unusual program behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(X) Demonstrate the ability to compromise
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Handle difficult people
(X) Accept criticism

*Learning objectives are specifically for Software Development, not a complete list.

# CPSC 294 C PROGRAMMING LANGUAGE

## COURSE OUTLINE

The objectives of the course should be taught with equal emphasis. Although a number of students will be taking the course just to learn C language, this is a three credit hour course and the other objectives are also important. Some complex data structures should be used in programming assignments. Advanced debugging tools such as a symbolic debugger should be used.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Boot and/or log on to a computer
(X) Use operating system commands
(X) Manage a file
(X) Name a file
(X) Create and use subdirectories
(X) Create command files
(X) Backup & restore hard drives
(X) Format disks
(X) Use terminal emulator
(X) Use a text editor
(X) Use an interpreter
(X) Use a compiler
(X) Use an assembler
(X) Use a linker
(X) Use a symbolic debugger
(X) Use assembly language
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Use documentation to learn new software
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Select appropriate system architecture
(X) Select appropriate design notation
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Use meaningful variables
(X) Select appropriate data structures
(X) Use number systems
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write program comments
(X) Construct subprograms
(X) Use selection (branching) methods
(X) Use iteration (looping) methods
(X) Write I/O (Input/Output) commands
(X) Write system calls
(X) Convert data among number systems
(X) Represent data in various data types
(X) Write code to a specified style

(X) Use simple data types
(X) Use arrays
(X) Use records
(X) Use files
(X) Use pointers
(X) Use linked lists
(X) Use trees
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Identify unusual program behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(X) Demonstrate the ability to compromise
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Handle difficult people
(X) Accept criticism

*Learning objectives are specifically for Software Development, not a complete list.

# CSFT 223 REAL TIME PROGRAMMING

## COURSE DESCRIPTION

Basic theoretical methods of information processing are presented. Includes methods of signal representation, data conversion, decision making, filtering, digital error problems, machine interaction and process control applications.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Use a terminal emulator
(X) Use a compiler
(X) Use a symbolic debugger
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Apply cross-referencing techniques
(X) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(X) Read and write interface documents for hardware, software and data
(X) Select appropriate system architecture
(X) Select appropriate design notation
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Create and use a modular design
(X) Select appropriate data structures
(I) Apply memory management techniques
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write system calls
(X) Convert data among number systems
(X) Write code to a specified style
(X) Use linked lists
(X) Use trees
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Modify program behaviors
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Write user documentation to specified format
(X) Schedule and use time effectively
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Accept criticism

# CSFT 224 SOFTWARE TEST TECHNIQUES

## COURSE DESCRIPTION

Introduction to software testing methodologies. Emphasis on program development techniques which aid testing. Introduction to proof of correctness and completeness. Superficial look at validation.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Create command files
(X) Use a terminal emulator
(X) Use a compiler
(X) Use an assembler
(X) Use a linker
(X) Use a symbolic debugger
(X) Use at least one high-level language
(X) Use documentation to learn new software
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(X) Read and write interface documents for hardware, software and data
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write system calls
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(I) Test for functional correctness and completeness
(I) Apply intermediate testing techniques
(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Read and analyze code
(X) Interpret formal specification changes
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(I) Write test plans
(X) Schedule and use time effectively
(X) Function as effective team member
(X) Understand importance of quality
(X) Handle difficult people
(X) Accept criticism

# CIS 262 COMPUTER GRAPHICS SYSTEMS

## COURSE DESCRIPTION

Concepts and principles, survey of preset display and input techniques, systems and applications. Study of basic concepts, mathematical and geometric principles, design and use of graphics software packages, design and implementation of an application using available hardware and software.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Follow documentation instructions
(X) Use documentation to learn new software
(X) Represent data in various data types
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Modify program behavior

*Learning objectives are specifically for Software Development, not a complete list.

# CSFT 232 ADVANCED SOFTWARE ENGINEERING

## COURSE DESCRIPTION

Advanced strudy of large program design and documentation, group problems, software maintenance and reliability.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Use operating system commands
(X) Manage a file
(X) Use a compiler
(X) Use at least one high-level language
(X) Follow documentation instructions
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(X) Read and write interface documents for hardware, software and data
(X) Select appropriate system architecture
(X) Select appropriate design notation
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Create and use a modular design
(X) Apply memory management techniques
(X) Adapt to styles of others
(X) Interpret designs of others
(X) Write program comments
(X) Write system calls
(X) Convert data among number systems
(X) Represent data in various data types
(X) Write code to a specified style
(X) Constructively critique code from others
(X) Identify and test critical points
(X) Test for logical correctness
(X) Test for functional correctness and completeness
(X) Apply intermediate testing techniques
(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Implement formal specification changes
(X) Interpret formal specification changes
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Give a program design presentation
(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(I) Use public speaking skills
(X) Write user documentation to specified format

(X) Write a software requirements document(SRD)
(X) Rough-out a program reference manual
(X) Rough-out a user guide
(X) Write test plans
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(X) Demonstrate the ability to compromise
(X) Demonstrate leadership techniques
(X) Use tact
(X) Function as effective team member
(X) Understand importance of quality
(X) Demonstrate dependability
(X) Handle difficult people
(X) Accept criticism

# CSFT 233 ADVANCED ASSEMBLY LANGUAGE PROGRAMMING

## COURSE DESCRIPTION

Program design and practice with assembly languages, macro definitions, conditioned assembly, advanced I/O, floating point operations.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Use operating system commands
(X) Manage a file
(X) Use a text editor
(X) Use an assembler
(X) Use a linker
(X) Use a symbolic debugger
(X) Use assembly language
(X) Apply cross-referencing techniques
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write HIPO(Hierarchy of Input, Processing, and Output)
(X) Read and write interface documents for hardware, software and data
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Create and use a modular design
(X) Apply memory management techniques
(X) Write program comments
(X) Write system calls
(X) Convert data among number systems
(X) Represent data in various data types
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Test for functional correctness and completeness
(X) Apply intermediate testing techniques
(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts
(X) Rough-out a user guide
(X) Write test plans
(X) Demonstrate punctuality
(X) Schedule and use time effectively
(X) Understand importance of quality

# CIS 270 NETWORKING & TELECOMMUNICATIONS

## COURSE DESCRIPTION

This course reviews data, text, graphics, and voice communications technology and their applications. Included is vocabulary, configuration of local networks, modems, rates, and standards. An overview of protocols is given.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Identify and use computer terminology
(X) Use operating system commands
(X) Manage a file
(X) Follow documentation instructions
(X) Use documentation to learn new software
(X) Write system calls
(X) Read and analyze code
(X) Use basic architecture concepts
(X) Use active listening skills
(X) Use good questioning techniques
(X) Schedule and use time effectively

*Learning objectives are specifically for Software Development, not a complete list.

# CSFT 235 LISP PROGRAMMING

## COURSE DESCRIPTION

Syntax and semantics of LISP programming language, style, recursion, tail recursion, algorithm development, list processing techniques.

## LEARNING OBJECTIVES

After completing this course, the student should be able to

(X) Use operating system commands
(X) Manage a file
(X) Use a text editor
(X) Use an interpreter
(X) Use a compiler
(X) Use a symbolic debugger
(X) Use documentation to learn new software
(X) Read and write pseudo code
(X) Read and write flow charts
(X) Read and write data flow diagrams
(X) Read and write structure charts
(X) Read and write a top-down design
(X) Write stepwise refinement
(X) Write program comments
(X) Write system calls
(X) Identify and test critical points
(X) Test all branch possibilities
(X) Test for invalid input
(X) Test for logical correctness
(X) Test for functional correctness and completeness
(X) Apply intermediate testing techniques
(X) Identify unusual program behavior
(X) Clearly document unusual behavior
(X) Read and analyze code
(X) Modify program behaviors
(X) Understand importance of quality
(X) Demonstrate dependability

# ELEC 131 BASICS OF ELECTRONICS

## COURSE DESCRIPTION

Topics discussed will provide for a systematic approach to electricity/electronic theory. Terminology, circuit layout, instrumentation applications, functions of systems will be covered. An introduction to printed circuit board design, preparation, and processing will be covered including hook-up wiring and interconnection techniques.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(I) Run hardware diagnostic programs
(X) Use basic external and internal hardware concepts
(X) Use basic architecture concepts

*Learning objectives are specifically for Software Development, not a complete list.

# ENGL 151 COMPOSITION/RHETORIC I

## COURSE DESCRIPTION

A beginning freshman course in writing. Development of paragraphs and the whole composition, study of model essays, extensive theme writing, individual conferences, departmental competency exam. Assessment prior to enrollment recommended.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(I) Write memoranda
(I) Write business letters
(X) Schedule and use time effectively

*Learning objectives are specifically for Software Development, not a complete list.

# MATH 181 COLLEGE ALGEBRA

## COURSE DESCRIPTION

A study of relations and functions, including linear, polynomial, rational, exponential, and logarithmic, inverse functions, composition of functions, absolute value, variation, theory of equations, complex numbers, systems of equations, matrices, progressions, conic sections, and the binomial theorem.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Write stepwise refinement
(X) Convert data among number systems
(X) Use number systems
(X) Test for logical correctness

*Learning objectives are specifically for Software Development, not a coplete list.

# PSYC 121 APPLIED PSYCHOLOGY

## COURSE DESCRIPTION

A survey of the applications of psychological knowledge and methodology in the fields of business, industry, education, medicine, law enforcement, and government work. Emphasis on group dynamics and adjustment factors for employment and advancement.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(I) Exhibit ethical behavior
(X) Demonstrate leadership techniques
(I) Demonstrate knowledge of basic social graces
(X) Use tact

*Learning objectives are specifically for Software Development, not a complete list.

# SPCM 151 FUNDAMENTALS OF SPEECH COMMUNICATION

## COURSE DESCRIPTION

Survey of basic factors affecting human interaction through communication; emphasis on the development of oral communication competencies; practice in delivering oral presentations.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Use active listening skills
(X) Use good questioning techniques
(X) Defend an idea
(X) Use public speaking skills
(I) Use good telephone skills

*Learning objectives are specifically for Software Development, not a complete list.

# ECON 121 INTRODUCTION TO ECONOMICS

## COURSE DESCRIPTION

This course is a study of economics of current issues including antitrust, deregulation, social security, labor, and the banking system. Also included are the purpose, functions, and results of a capitalistic system.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Use active listening skills
(X) Write business letters
(X) Demonstrate ethical behavior

*Learning objectives are specifically for Software Development, not a complete list.

# HUM 151 INTRO TO HUMANITIES

## COURSE DESCRIPTION

A course designed to achieve a clearer understanding of the nature of man and his need to create. The course explores the relationship between one's own values, feelings, attitudes and ideas and man's cultural achievements.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(I) Exhibit appropriate dress and grooming
(X) Exhibit ethical behavior
(X) Demonstrate knowledge of basic social graces
(X) Use tact

*Learning objectives are specifically for Software Development, not a complete list.

# MATH 182 TRIGONOMETRY

## COURSE DESCRIPTION

A study of angular measure, functions of angles, identities, solution of triangles, equations, inverse trigonometric functions, complex numbers, and polar coordinates.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(X) Use number systems
(X) Write stepwise refinement
(X) Convert data among number systems
(X) Test for logical correctness

*Learning objectives are specifically for Software Development, not a complete list.

# ENGL 291 TECHNICAL WRITING

## COURSE DESCRIPTION

A comprehensive introduction to technical writing and technical communication. Review of mechanical techniques employed in correct technical writing. Preparation of reports, proposals, technical papers, abstracts, and summaries within the specific areas of technical interest of the student.

## LEARNING OBJECTIVES*

After completing this course, the student should be able to

(I) Use a word processing package
(X) Follow documentation instructions
(X) Write program comments
(X) Interpret formal specification changes
(X) Write user documentation to specified format
(X) Write a software requirements document(SRD)
(X) Rough-out a program reference manual
(X) Rough-out a user guide
(X) Write test plans

*Learning objectives are specifically for Software Development, not a complete list.

# Index


C

Committee
size . . . . . 1-5
Competency
duties . . . . . 2-35
Course descriptions
catalog . . . . . 3-57

D

DACUM
facilitator . . . . . 2-31
final report . . . . . 2-31
Handbook . . . . . 2-31
panel . . . . . 2-31
panel members . . . . . 2-31
recorder . . . . . 2-31
Degree plans . . . . . 3-53
Duties . . . . . 2-35

G

General education core . . . . . 3-54
General survey . . . . . 1-2

J

Job description . . . . . 2-35
Job titles . . . . . 2-35

N

Needs analysis . . . . . 1-2

S

Software development
degree plan . . . . . 3-55
Summary
Executive . . . . . 1-6
Survey
business . . . . . 1-8
general . . . . . 1-2
mean . . . . . 1-11
median . . . . . 1-11
mode . . . . . 1-11
results . . . . . 1-10
software technology . . . . . 1-10
Survey statistics
Ada Programming . . . . . 1-15
Algebra . . . . . 1-17

Assemble Language Programming . . . . . 1-16
Data Structures . . . . . 1-19
Fortran-77 Programming . . . . . 1-20
Group Dynamics/Problem Solving . . . . . 1-21
Pascal Programming . . . . . 1-22
Physics I . . . . . 1-23
Real Time Programming . . . . . 1-24
Software Engineering . . . . . 1-25
Software Test Techniques . . . . . 1-26
Speech Communications . . . . . 1-27
Technical Writing . . . . . 1-28
Trigonometry . . . . . 1-29

T

Task inventory . . . . . 2-41
Task statements
instructions . . . . . 2-40
Tasks 2-35
Technical program core . . . . . 3-54

ERIC Clearinghouse for
Junior Colleges DEC 16 1988